SIMULINK (1)

快速入门

本章主要内容和学习目的

首先介绍计算机仿真技术和仿真建模方法的基本概念，以便对建模和仿真有个初步和整体的认识；然后对 **Simulink** 进行简单介绍，并以一个简单例子进行引导；最后介绍 **Simulink** 的工作原理，为后续的深入掌握 **Simulink** 打下基础。

- 动态系统的计算机仿真
- 仿真三要素
- **Simulink** 与建模仿真
- **Simulink** 的安装
- 创建一个简单模型
- 模型基本结构

1.1 动态系统的计算机仿真

1.1.1 系统与模型

1. 系统

系统只指具有某些特定功能、相互联系、相互作用的元素的集合。这里的系统是指广义上的系统，泛指自然界的一切现象与过程，例如工程系统如控制系统、通讯系统等，非工程系统如股市系统、交通系统、生物系统等。

2. 系统模型

系统模型是对实际系统的一种抽象，是对系统本质（或是系统的某种特性）的一种描述。模型具有与系统相似的特性。好的模型能够反映实际系统的主要特征和运动规律。

模型可以分为实体模型和数学模型。

实体模型又称物理效应模型，是根据系统之间的相似性而建立起来的物理模型，如建筑模型等。

数学模型包括原始系统数学模型和仿真系统数学模型。原始系统数学模型是对系统的原始数学描述。仿真系统数学模型是一种适合于在计算机上演算的模型，主要是指根据计算机的运算特点、仿真方式、计算方法、精度要求将原始系统数学模型转换为计算机程序。

<table>
<tr><td>静态系统模型</td><td colspan="3">动态系统模型</td></tr>
<tr><td rowspan="3">代数方程</td><td colspan="2">连续系统模型</td><td>离散系统模型</td></tr>
<tr><td>集中参数</td><td>分布参数</td><td rowspan="2">差分方程</td></tr>
<tr><td>微分方程</td><td>偏微分方程</td></tr>
</table>

1.1.2 计算机仿真

1. 仿真的概念

仿真是以相似性原理、控制论、信息技术及相关领域的有关知识为基础，以计算机和各种专用物理设备为工具，借助系统模型对真实系统进行试验的一门综合性技术。

2. 仿真分类

（1）实物仿真：又称物理仿真。是指研制某些实体模型，使之能够重现原系统的各种状态。早期的仿真大多属于这一类。

优点：直观，形象，至今仍然广泛应用。

缺点：投资巨大、周期长，难于改变参数，灵活性差。

（2）数学仿真：是用数学语言去描述一个系统，并编制程序在计算机上对实际系统进行研究的过程。

优点：灵活性高，便于改变系统结构和参数，效率高（可以在很短时间内完成实际系统很长时间的动态演变过程），重复性好

缺点：对某些复杂系统可能很难用数学模型来表达，或者难以建立其精确模型，或者由于数学模型过于复杂而难以求解

（3）半实物仿真：又称数学物理仿真或者混合仿真。为了提高仿真的可信度或者针对一些难以建模的实体，在系统研究中往往把数学模型、物理模型和实体结合起来组成一个复杂的仿真系统，这种在仿真环节中存在实体的仿真称为半物理仿真或者半物理仿真，如飞机半实物仿真等。

3. 计算机仿真

计算机仿真是在研究系统过程中根据相似性原理，利用计算机来逼真模拟研究系统。研究对象可以是实际的系统，也可以是设想中的系统。在没有计算机以前，仿真都是利用实物或者它的物理模型来进行研究的，即物理仿真。物理仿真的优点是直接、形象、可信，缺点是模型受限、易破坏、难以重用。计算机仿真可以用于研制产品或设计系统的全过程，包括方案论证、技术指标确定、设计分析、故障处理等各个阶段。如训练飞行员、宇航员的方针工作台和仿真机舱等。

1.2 仿真的三要素

计算机仿真的三个基本要素是系统、模型和计算机，联系着它们的三项基本活动是模型建立、仿真模型建立（又称二次建模）和仿真试验。

数学仿真采用数学模型，用数学语言对系统的特性进行描述，其工作过程是：

- 建立系统的数学模型；
- 建立系统仿真模型，即设计算法，并转化为计算机程序，使系统的数学模型能为计算机所接受并能在计算机上运行；
- 运行仿真模型，进行仿真试验，再根据仿真试验的结果进一步修正系统的数学模型和仿真模型。

1.3 Simulink 与建模仿真

1.3.1 Simulink

Simulink 是一种用来实现计算机仿真的软件工具。它是**MATLAB** 的一个附加组件，可用于实现各种动态系统（包括连续系统、离散系统和混合系统）的建模、分析和仿真。

特点：易学易用，能够依托**MATLAB**提供的丰富的仿真资源

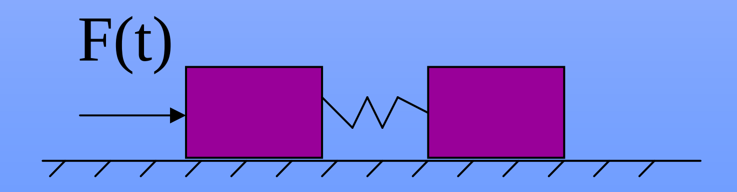
1.3.2 Simulink 的应用领域

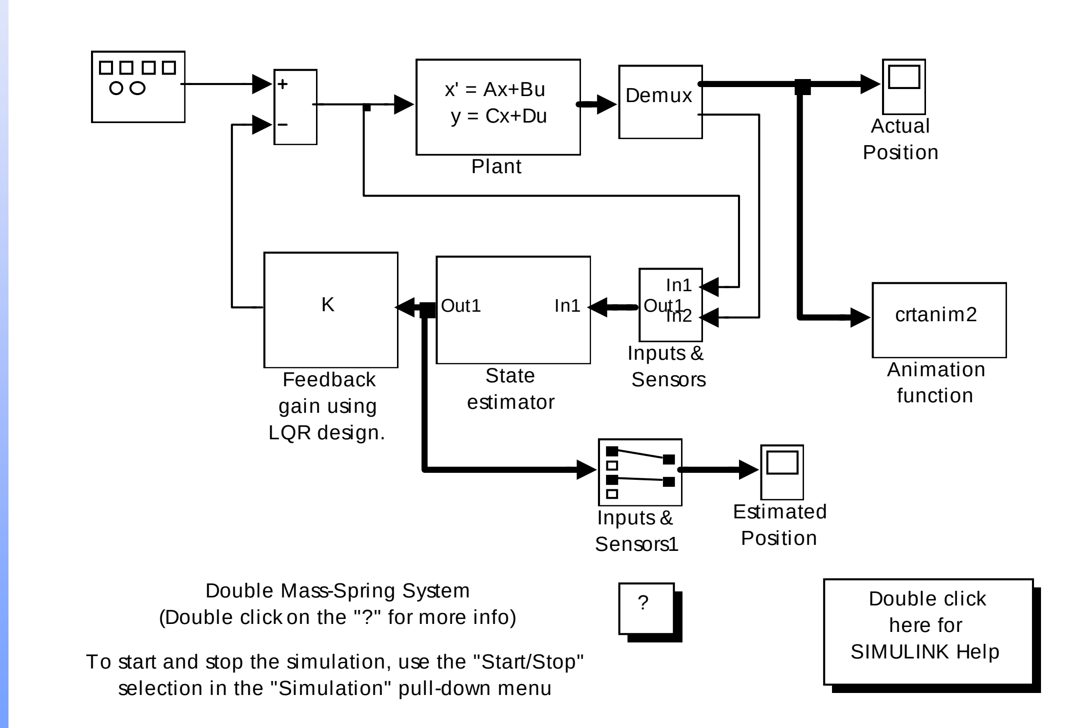
（**1**）通讯与卫星系统；
（**2**）航空航天系统；
（**3**）生物系统；
（**4**）船舶系统；
（**5**）汽车系统；
（**6**）金融系统；
（**7**）控制系统。

1.3.3 Simulink 应用举例 （原教材P6例子）

在Matlab命令窗口中输入 dblcart1

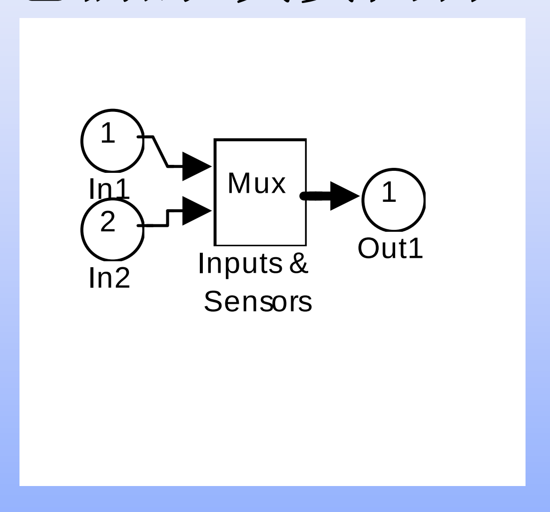
右图所示的模型用来模拟双质量－弹簧系统在光滑平面上受一个周期力情况下的运动状态，其中周期力只作用在左边的质量块上。

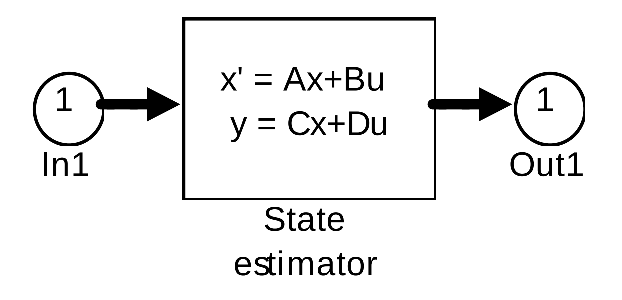




此模型中使用了状态 判断和**LQR**控制。

模型中还有几个“隐藏了真实身份”的子系统，如图中的**Inputs&Sensors** 模块和 **State estimator** 模块。 双击后可看到它们的“真实面目”。





x' = Ax+Bu
y = Cx+Du
Plant
Demux
Actual
Position
K
Feedback
gain using
LQR design.
Out1
In1
State
estimator
In1
Out1
In2
Inputs &
Sensors
crtanim2
Animation
function
Inputs &
Sensors1
Estimated
Position

Double Mass-Spring System
(Double click on the "?" for more info)

To start and stop the simulation, use the "Start/Stop"
selection in the "Simulation" pull-down menu

?

Double click
here for
SIMULINK Help

运行菜单选项【**Simulation>Start**】，则屏幕上出现双质量—弹簧系统运动状态的动画图形。

模型中的**Actural Position**模块和**Estimated Position**模块用来显示在一个周期内的左边质量块的位置轨迹。

1.4 Simulink 的安装

系统要求:

奔腾100以上CPU，16MB以上内存，Windows 9x或Windows NT

安装:

随MATLAB安装或单独安装。

启动 Simulink:

首先启动MATLAB，然后在MATLAB窗口中单击上面的Simulink按钮或在命令窗口中输入simulink。

1.5 创建一个简单模型

两个例子

例子1

$$\begin{cases} \dot{x}(t) = \sin(t) \\ x(0) = 0 \end{cases}$$

$$x(t) = -\cos(t) + C$$

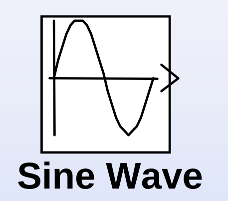
C 为常数 利用初始条件 → $C = 1$

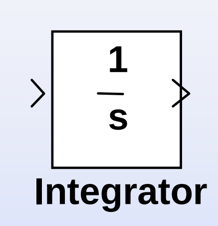
$$x(t) = -\cos(t) + 1$$

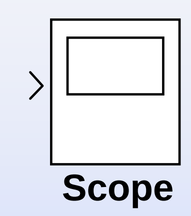
创建模型步骤

$$\begin{cases} \dot{x}(t) = \sin(t) \\ x(0) = 0 \end{cases} \qquad x(t) = -\cos(t) + 1$$

步骤一：添加模块

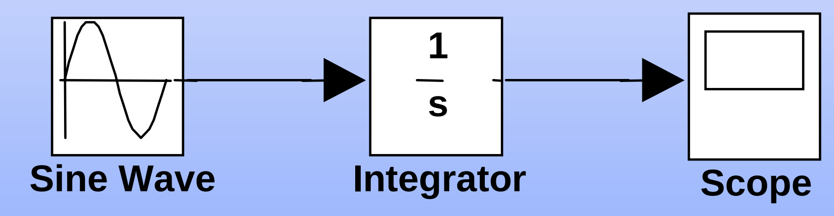






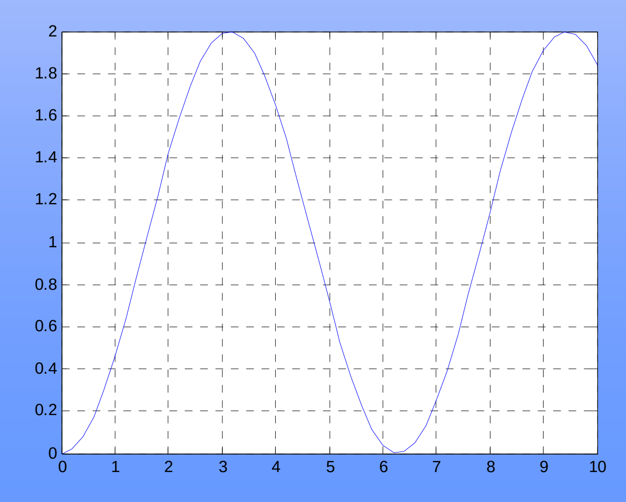
从源模块库（**Sources**）中复制正弦波模块（**Sine Wave**）。
连续模块库（**Continuous**）复制积分模块（**Integrator**）。
输出显示模块库（**Sinks**）复制示波器模块（**Scope**）。

步骤二：连接模块



步骤三：运行仿真

双击示波器模块，打开**Scope**窗口。双击模型窗口菜单中的【 **Simulation>Start** 】，仿真执行，结果如图所示。

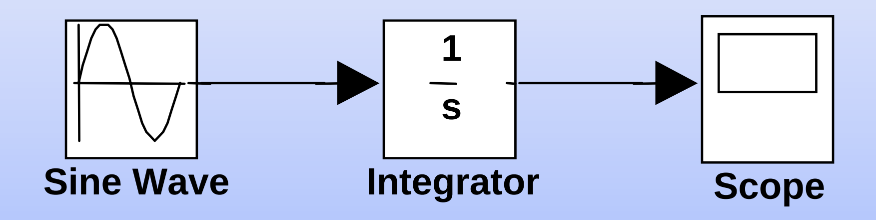


如果将以上算例的初始条件改为： $x(0) = -1$

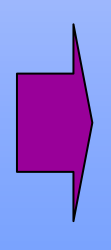
$$\begin{cases} \dot{x}(t) = \sin(t) \\ x(0) = -1 \end{cases} \xrightarrow{\text{利用初始条件}} C = 0$$

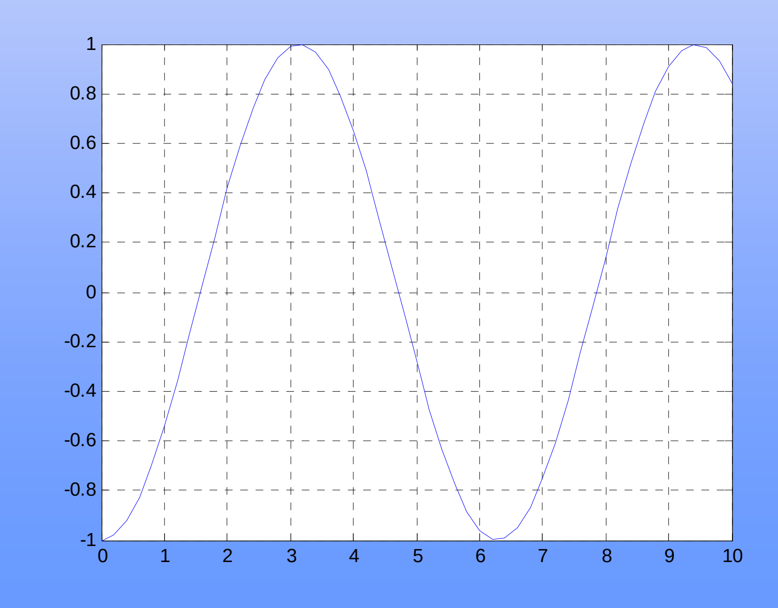
系统的解析解为： $x(t) = -\cos(t)$

Simulink模型：



在仿真时，需要将积分模块 $\frac{1}{s}$ 的初始值设置为－1，最终可以得到标准的余弦曲线





例子2

单自由度系统：

$$m\ddot{x}+c\dot{x}+kx=0$$

初始条件：

$$x(0)=x_0=1,\quad \dot{x}(0)=\dot{x}_0=0$$

要求：采用 **Simulink** 对系统进行仿真。已知参数：$m=1,\quad c=1,\quad k=1$

解析解为：

$$x(t)=e^{-\zeta\omega_0 t}\left[x_0\cos(\omega_d t)+\frac{\dot{x}_0+\zeta\omega_0 x_0}{\omega_d}\sin(\omega_d t)\right]$$

其中：

$$\omega_0=\sqrt{\frac{k}{m}}$$ ⟶ 系统固有频率

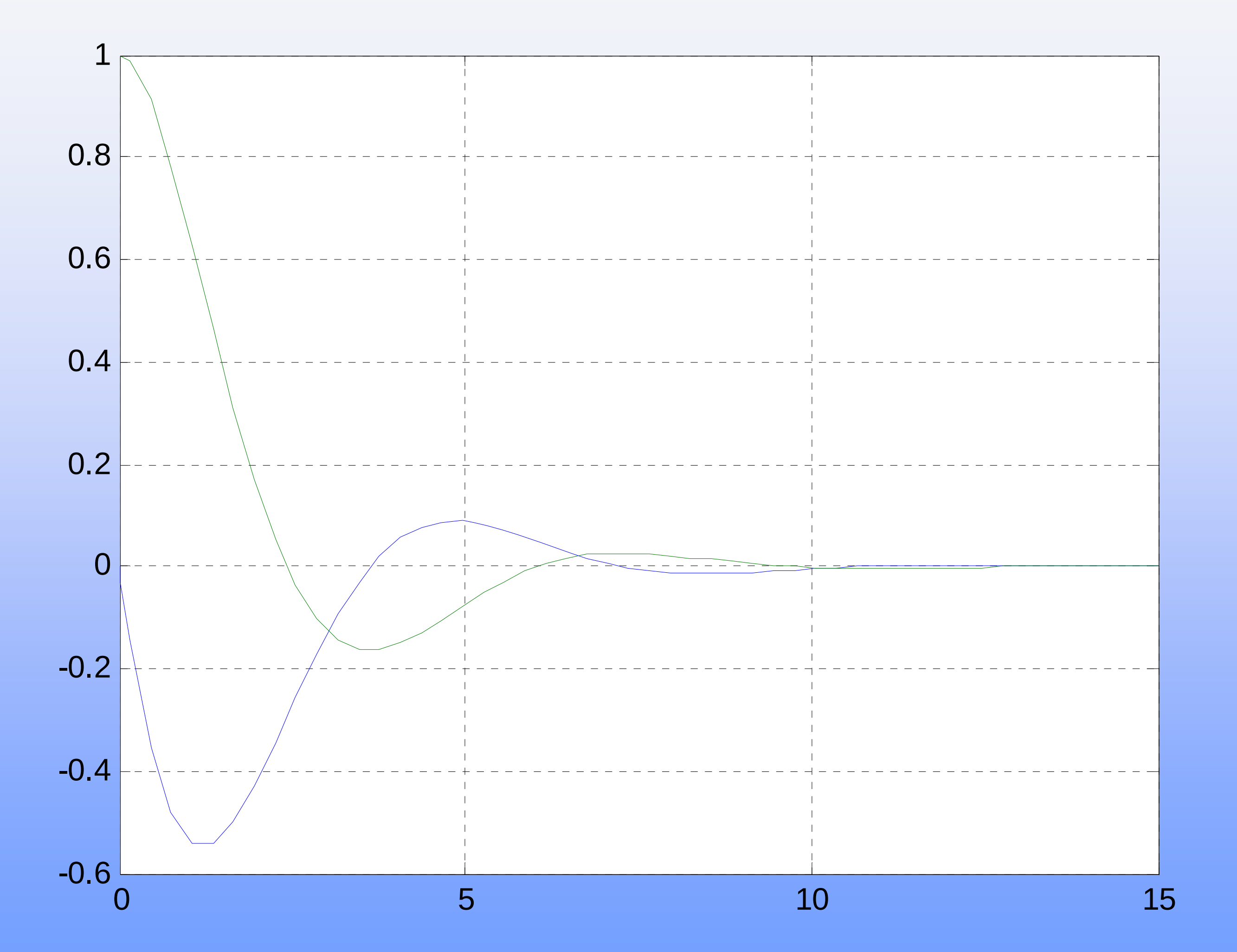
$$\omega_d=\omega_0\sqrt{1-\zeta^2}$$ ⟶ 阻尼固有频率

$$\zeta=\frac{c}{2\sqrt{km}}$$ ⟶ 相对阻尼系数

$$m\ddot{x} + c\dot{x} + kx = 0 \qquad x(0) = x_0 = 1, \quad \dot{x}(0) = \dot{x}_0 = 0$$

$$\ddot{x} + \frac{c}{m}\dot{x} + \frac{k}{m}x = 0$$

已知参数：$m = 1, \quad c = 1, \quad k = 1$



1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
5
10
15

如果系统中没有阻尼，则动力方程为：

$$m\ddot{x} + kx = 0$$

已知参数： $m = 1, \quad k = 1$

初始条件：

$$x(0) = x_0 = 1, \quad \dot{x}(0) = \dot{x}_0 = 0$$

解析解为：

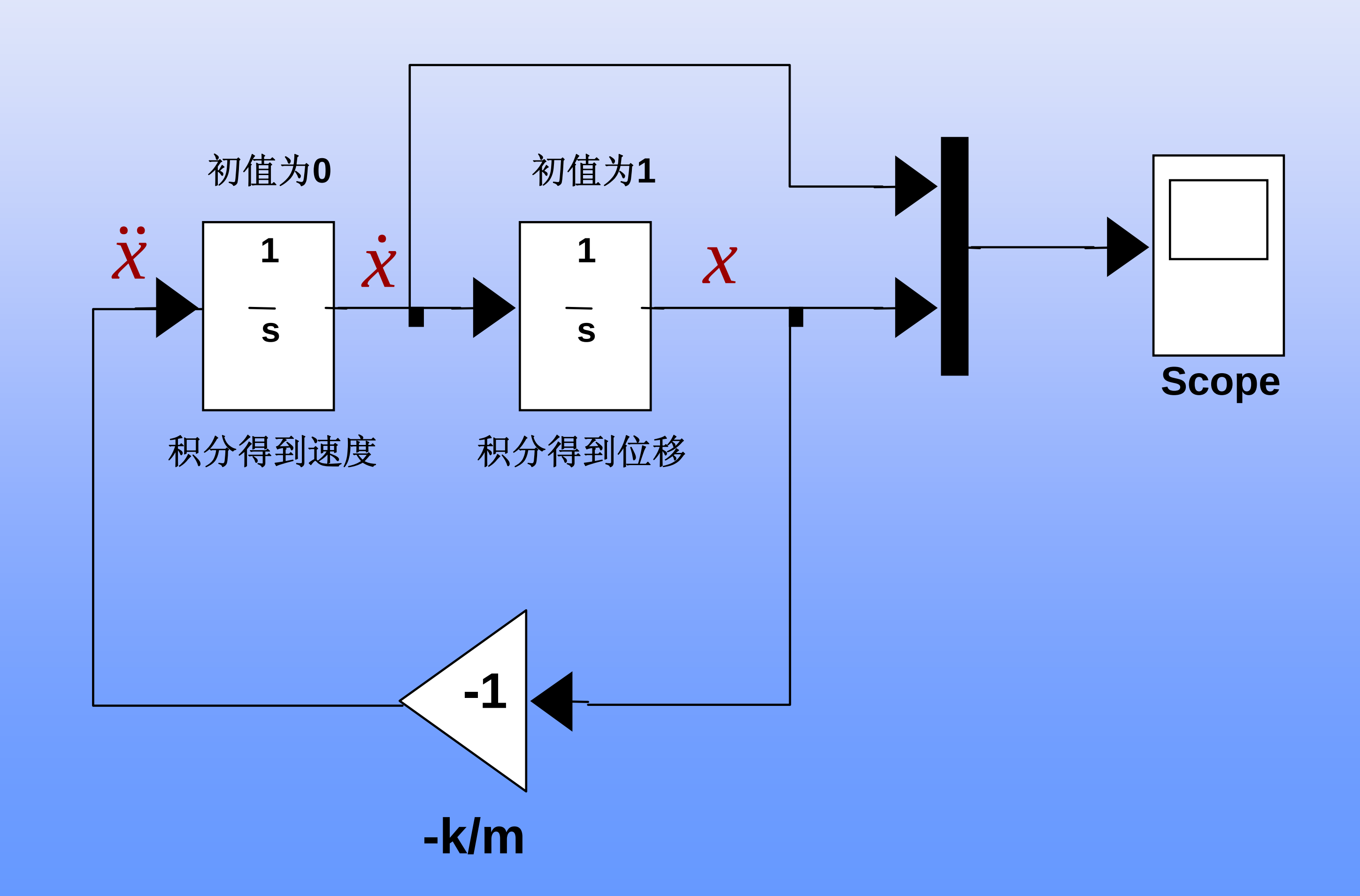
$$x(t) = x_0 \cos(\omega_0 t) + \frac{\dot{x}_0}{\omega_0} \sin(\omega_0 t)$$

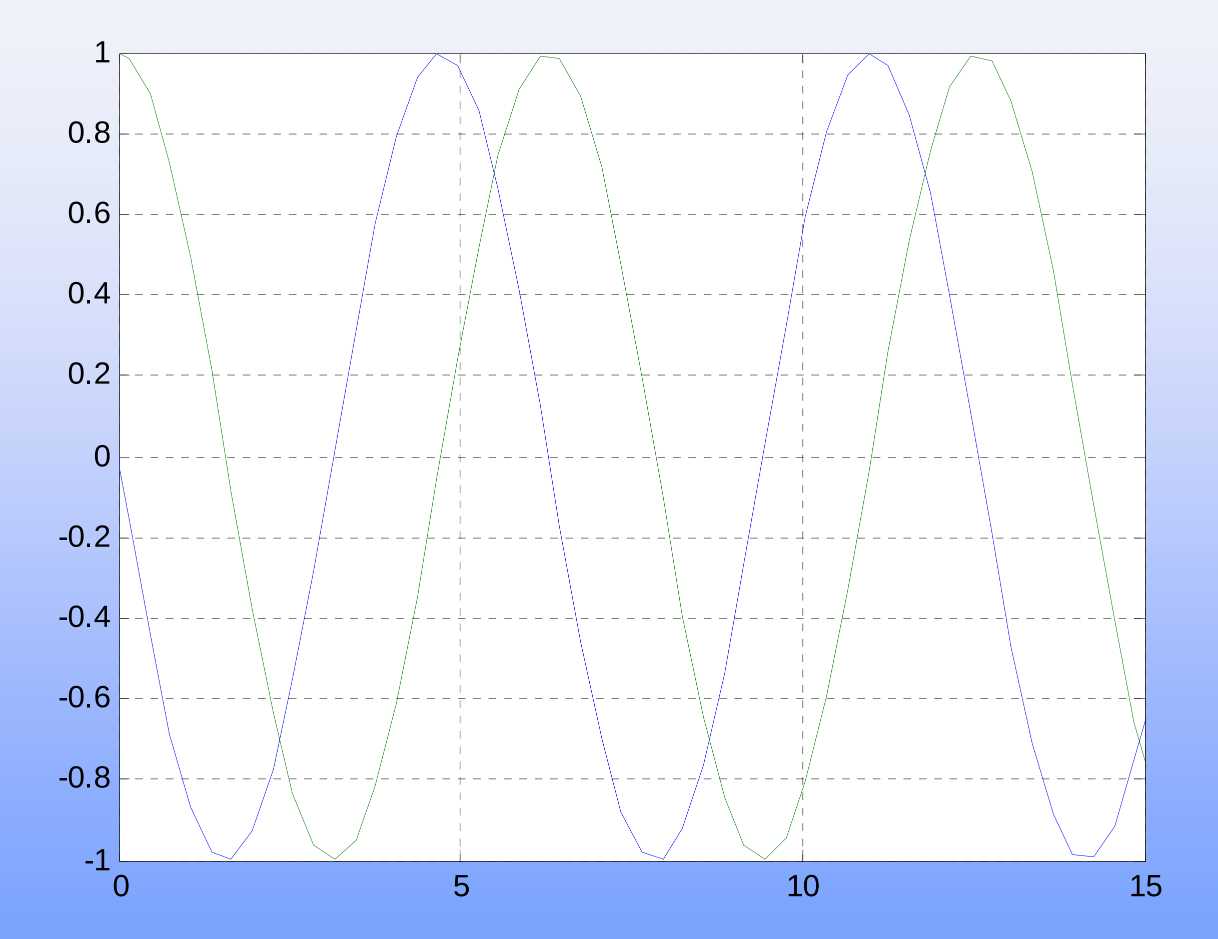
其中： $\omega_0 = \sqrt{\frac{k}{m}}$ ⟶ 系统固有频率

$$m\ddot{x} + kx = 0 \qquad \ddot{x} + \frac{k}{m}x = 0$$

已知参数： $m = 1, \quad k = 1$

初始条件： $x(0) = x_0 = 1, \quad \dot{x}(0) = \dot{x}_0 = 0$



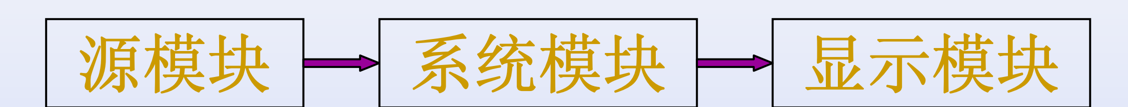


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
5
10
15

1.6 模型基本结构

一个典型的 **Simulink** 模型包括如下三种类型的元素：

- 信号源模块
- 被模拟的系统模块
- 输出显示模块



Simulink 模型元素关联图

- 信号源为系统的输入，它包括常数信号源、函数信号发生器（如正弦波和阶跃函数波等）和用户自己在 **MATLAB** 中创建的自定义信号。

- 系统模块作为中心模块是 **Simulink** 仿真建模所要解决的主要部分。

- 系统的输出由显示模块接收。输出显示的形式包括图形显示、示波器显示和输出到文件或 **MATLAB** 工作空间中三种。输出模块主要在 **Sinks** 库中。

SIMULINK (2)

建模方法

本章内容和学习目的

- 本章介绍一些模块的特殊操作，如模块操作、信号线操作、模型注释等，对模块和信号线有一个整体认识，以便为后续的动力学建模和仿真打下基础。
- 掌握模块操作、信号线操作、模型注释的基本技巧。

2.1 打开模型

2.2 模块操作

2.3 信号线操作

2.4 模型注释

2.5 创建一个复杂模型

2.6 模型和图形拷贝到**Word**文档中

2.1 打开模型

1. 新建模型

2. 打开一个已存在的模型

方法**1**：通过**【File>Open】**命令；

方法**2**：在 **MATLAB** 命令窗中直接键入模型名。

2.2 模块操作

2.2.1 调整模块大小

选中模块，模块四角出现小方块，然后按住鼠标拖曳

2.2.2 旋转模块

选中模块，然后选择菜单命令【Format>Rotate】，模块将顺时针方向旋转90度。

2.2.3 模块的内部复制

- 方法1：先按Ctrl键，再单击模块
- 方法2：右键拖曳模块
- 方法3：【Edit>Copy】和【Edit>Paste】
- 方法4：选中模块，Ctrl＋C复制，Ctrl＋V粘贴

2.2.4 删除模块

方法1：选中模块，按**Delete**键

方法2：选中模块，然后从模型窗口菜单中选择【**Edit>Clear**】

方法3：选中模块，然后选择菜单【**Edit>Cut**】，删至剪贴板

2.2.5 选中多个模块

方法1：按住**Shift**键，同时用鼠标单击想要选中的模块

方法2：使用“范围框”。在使用“范围框”后，若像继续选择模块，可按住**Shift**键，再单击要选的模块即可。

若多个模块被选中，则这些模块可以被当作一个整体进行操作，如移动、复制和删除等，操作方法和单个模块的操作方法相同。

2.2.6 改变模块的标签

在标签的位置上双击鼠标，则模块标签进行编辑状态。编辑完标签后，在标签外的任意位置上单击鼠标，则新的合法标签将被承认。

2.2.7 改变标签位置

选择菜单命令【**Format>Flip Name**】，模块的标签将发生翻转。若原标签位置在模块的左边，则翻转到右边。若原标签位置在模块的下方，则翻转到上方。

2.2.8 隐藏标签

选择菜单命令【**Format>Hide Name**】，则模块的标签从屏幕上消失。此时【**Hide Name**】变成为【**Show Name**】，选中它可以重新显示标签。

2.2.8 增加阴影

选择菜单命令【**Format>Show Drop Shadow**】，则模块增加阴影。此时【**Show Drop Shadow**】变成为【**Hide Drop Shadow**】，选中它可以隐藏阴影。

2.3 信号线操作

2.3.1 绘制信号线

2.3.2 移动线段

2.3.3 移动节点

2.3.4 删除信号线

2.3.5 分割信号线置

2.3.6 信号线标签

2.4 模型注释

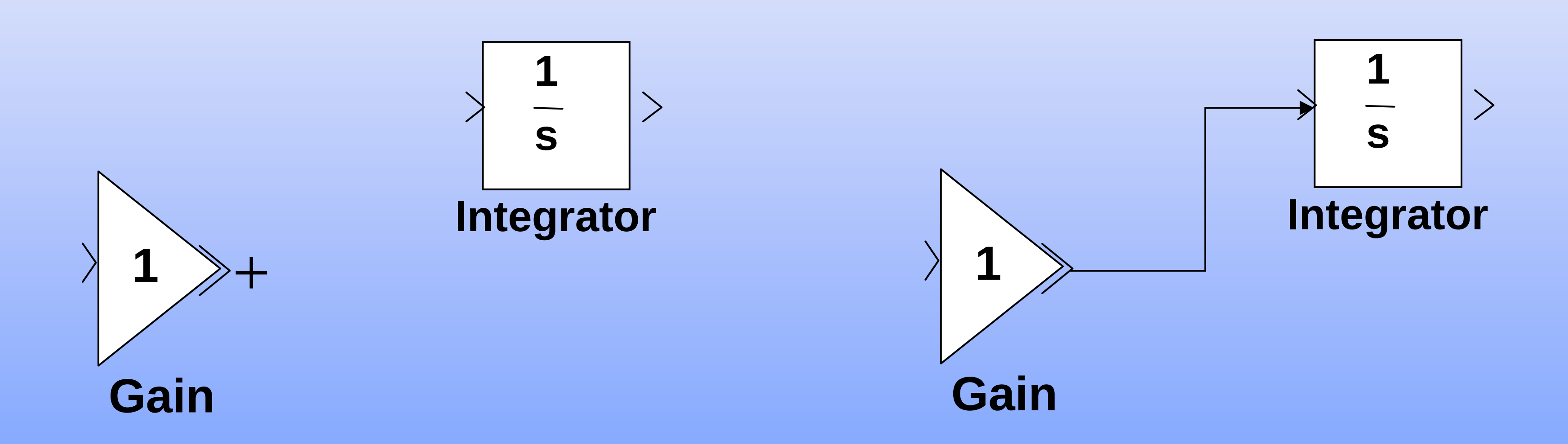
2.4.1 增加注释的方法

2.4.2 改变注释字体

注意：要养成及时对模块和信号线的命名以及及时对模型加标注的习惯，这会提高模型的可读性。

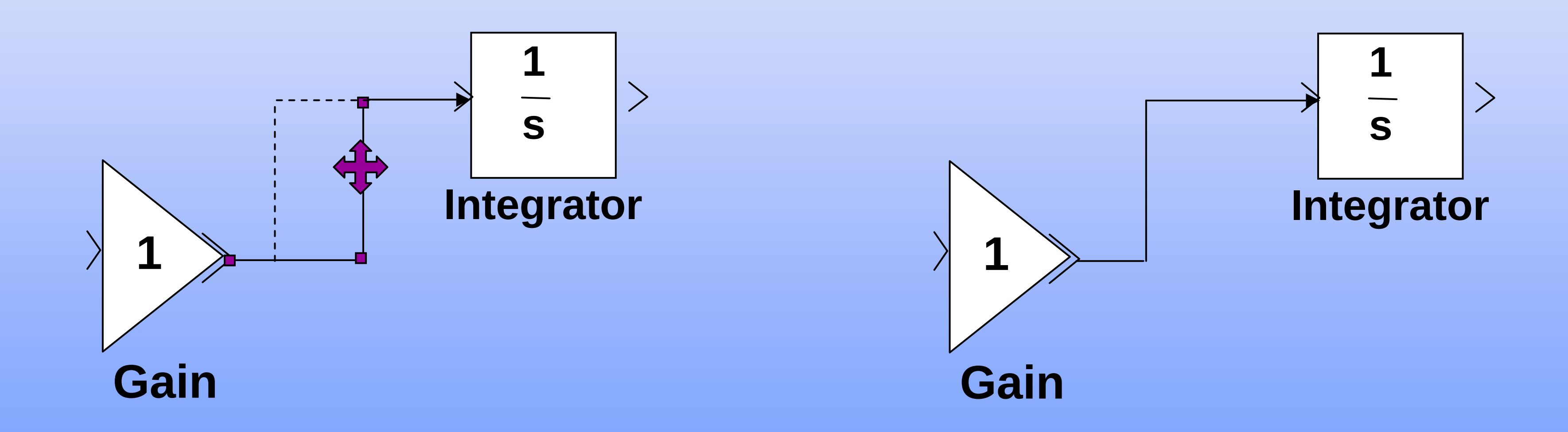
2.3.1 绘制信号线

- 由输出端口拖曳鼠标到输入端口，或拖曳鼠标由输入端口到输出端口



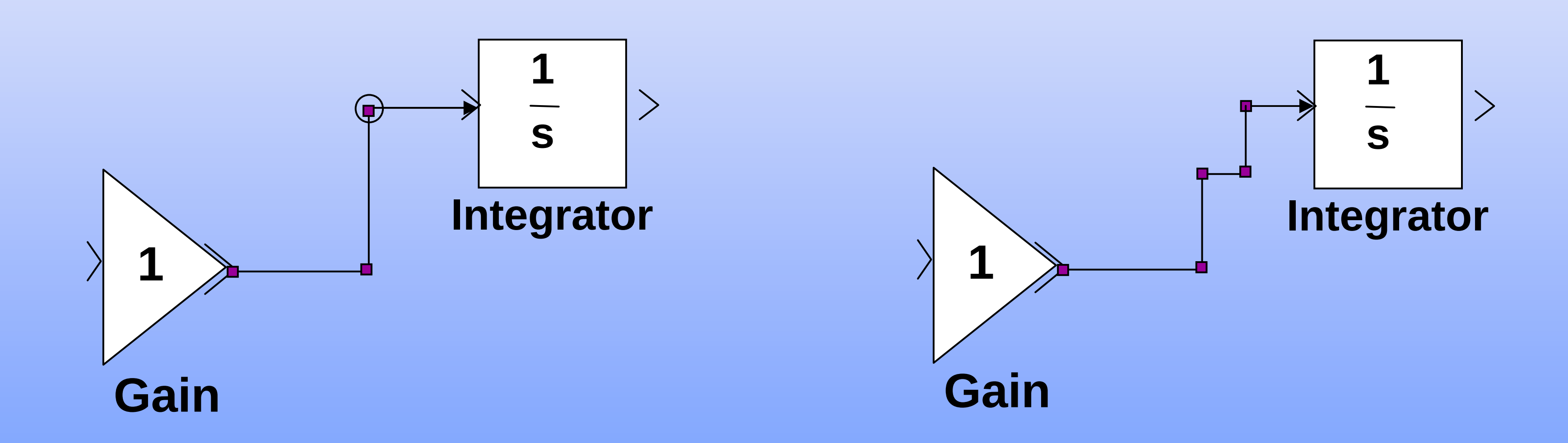
2.3.2 移动线段

- 若想移动信号线的某段，单击选中此段。移动鼠标到目标线段上，则鼠标的形状变为移动图标。按住鼠标，并拖曳到新位置。放开鼠标，则信号线被移动到新的位置。



2.3.3 移动节点

- 若想移动信号线的某段，单击选中此段。移动鼠标到目标线段上，则鼠标的形状变为移动图标。按住鼠标，并拖曳到新位置。放开鼠标，则信号线被移动到新的位置。

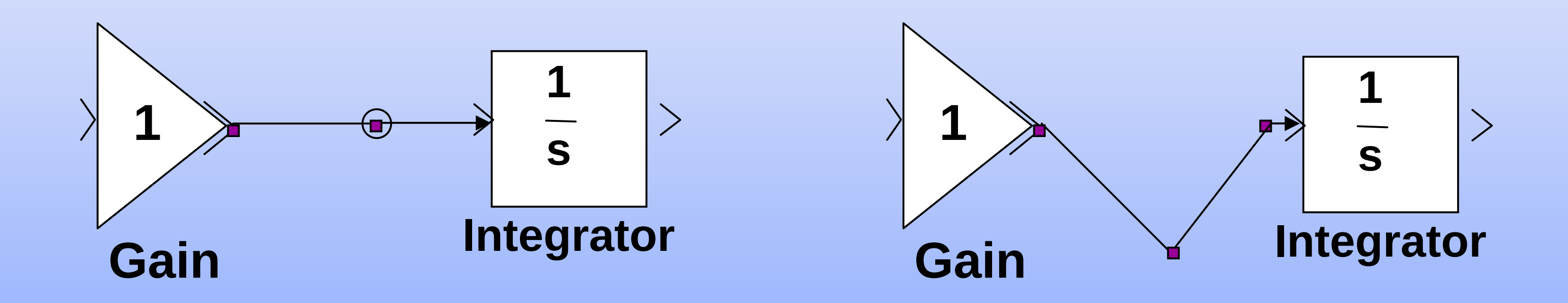


2.3.4 删除信号线

- 同删除模块一样，删除信号线可以选中信号线，然后按 **Delete 键**，或者利用菜单 **【Edit/Clear】**或**【Edit/Cut】**选项进行删除。

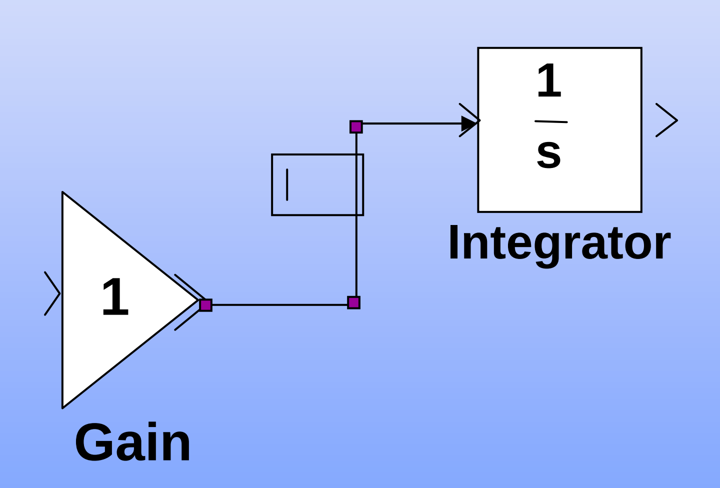
2.3.5 分割信号线置

- 先选中信号线。按住 **Shift** 键，然后在信号线上需要分割的点上单击鼠标，拖动节点到新的位置，放开鼠标即可。



2.3.6 信号线标签

- 每段信号线都可以有一个标签。双击要标注的信号线，则信号线的附近就会出现一个编辑区，在编辑区内输入标签的内容即可。



2.4 模型注释

使用模型注释可以使模型更易读懂，其效果如同 **MATLAB** 程序中的注释行一样。对于经常使用 **Simulink** 的用户，养成经常使用注释的习惯是非常重要的。

2.4.1 增加注释的方法

在模型窗口中的任何想要注释的部位上双击鼠标，将会出现一个编辑框，在该框内输入想要注释的内容即可。

2.4.2 改变注释字体

要改变注释内容的字体，先选中注释，选择模型窗口菜单中的【**Format Font**】选项，就会出现一个字体选择的对话框，选中认为合适的字体，然后按【**OK**】。

2.5 创建一个复杂模型

通过两个算例来说明建模中的其他一些技巧

例子1

一个生长在罐中的细菌的简单模型。假定细菌的出生率和当前细菌的总数成正比，死亡率和当前的总数的平方成正比。若以 x 代表当前细菌的总数，则细菌的出生率可表示为：

$$birth_rate = bx$$

细菌的死亡率可表示为：

$$death_rate = px^2$$

细菌总数的总变化率可表示为出生率与死亡率之差。因此系统可表示为如下的微分方程形式：

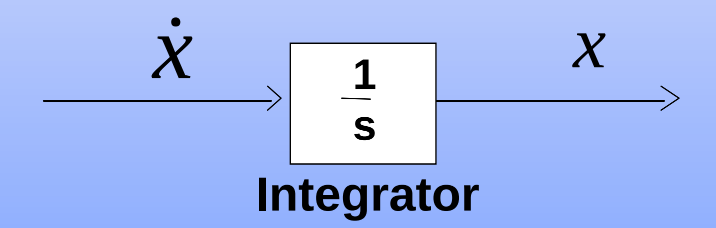
$$\dot{x} = bx - px^2$$

假定 $b = 1/h$，$p = 0.5/h$，当前细菌的总数为100，计算一个小时后罐中的细菌总数。

$$\dot{x} = bx - px^2$$

步骤一：添加模块

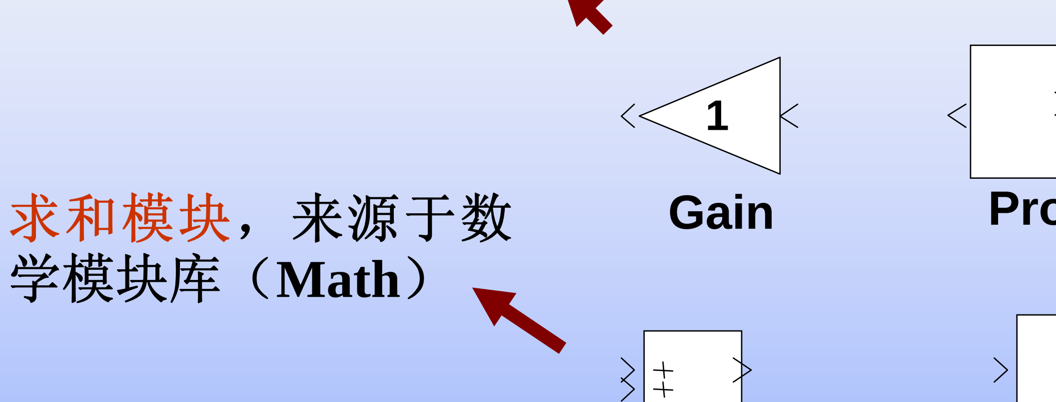
这是一个一阶系统，因此一个用来解微分方程的积分模块是必要的。积分模块的输入为 $\dot{x}$，输出为 x。

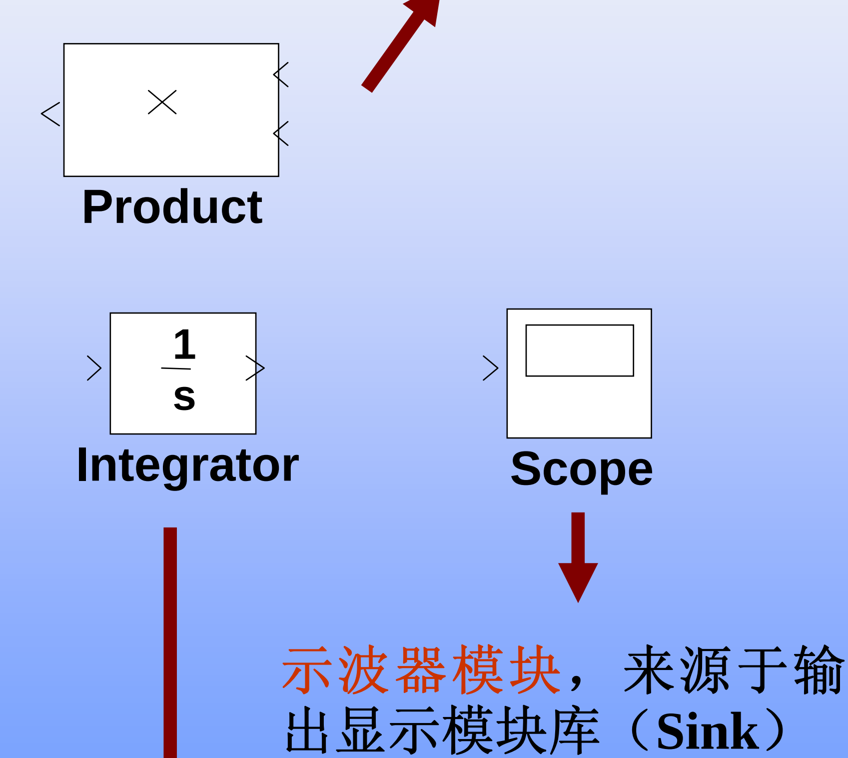


$$\dot{x} = bx - px^2$$

增益模块，来源于数学模块库（Math）

乘法模块，来源于数学模块库（Math），用于实现 x^2



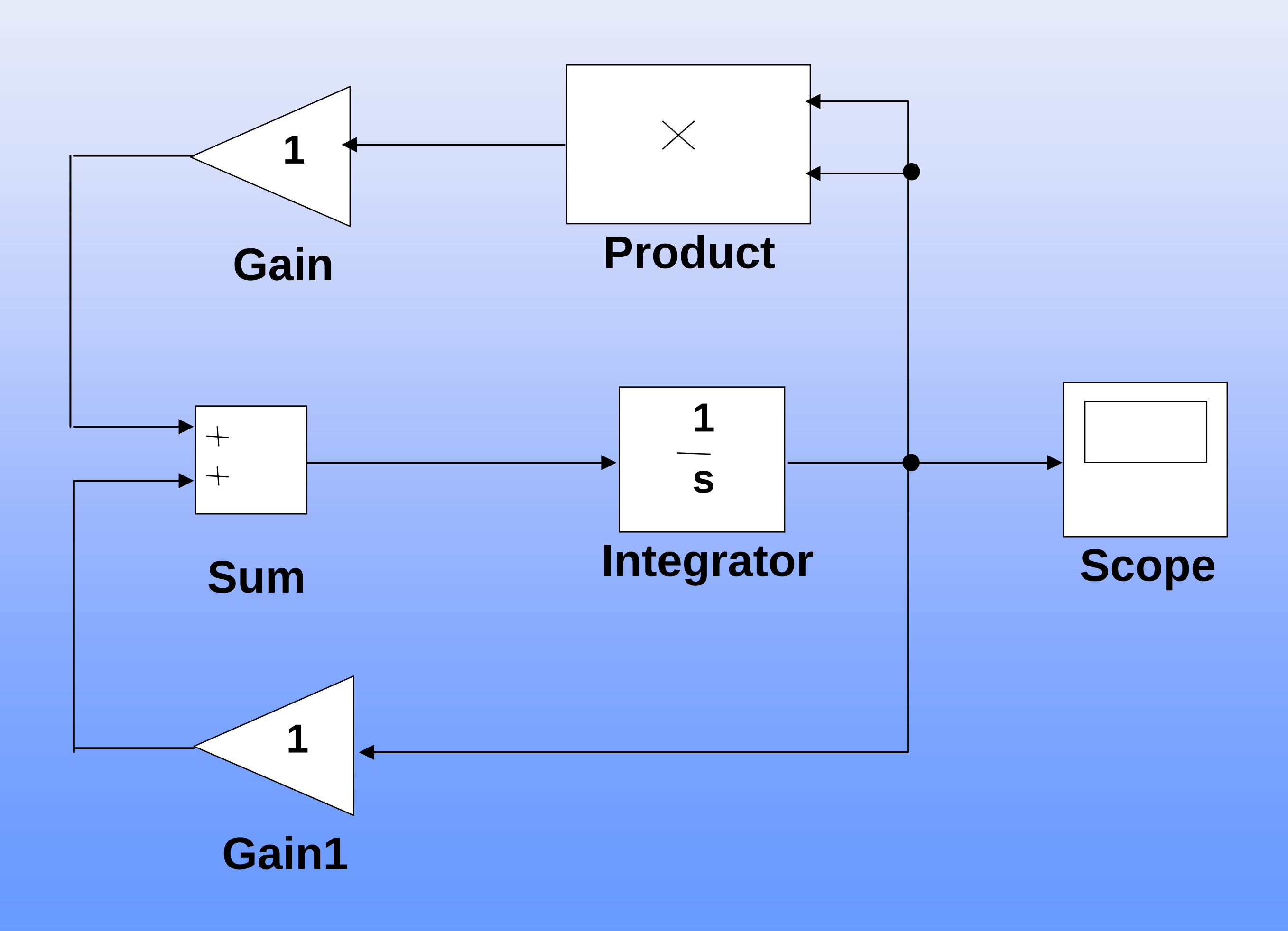


增益模块，来源于数学模块库（Math）

积分模块，来源于连续模块库（Continuous）

$$\dot{x} = bx - px^2$$

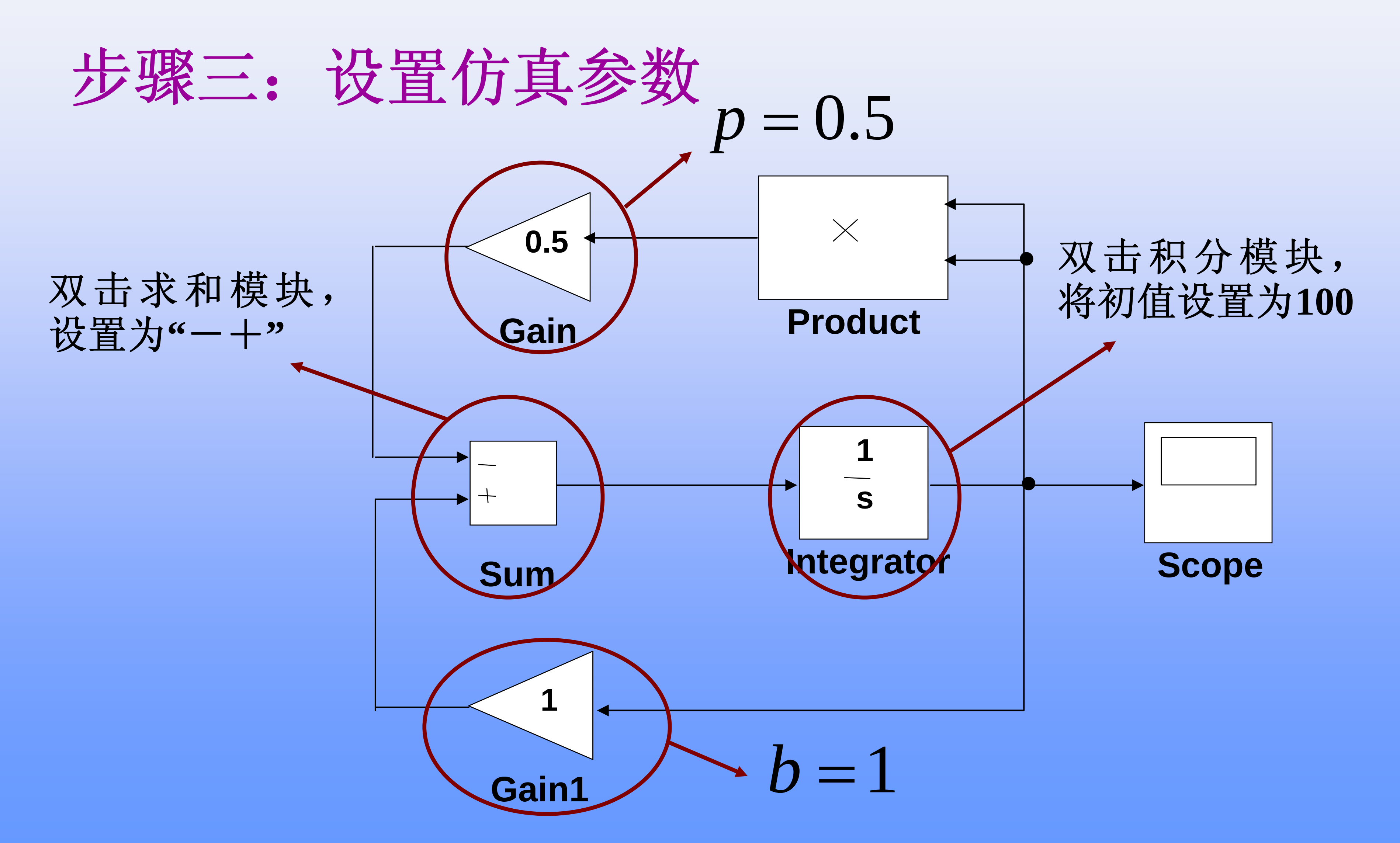
步骤二：连接模块



$$\dot{x} = bx - px^2 \qquad b = 1 \qquad p = 0.5$$

当前细菌的总数为100，计算一个消失后罐中的细菌总数

步骤三：设置仿真参数

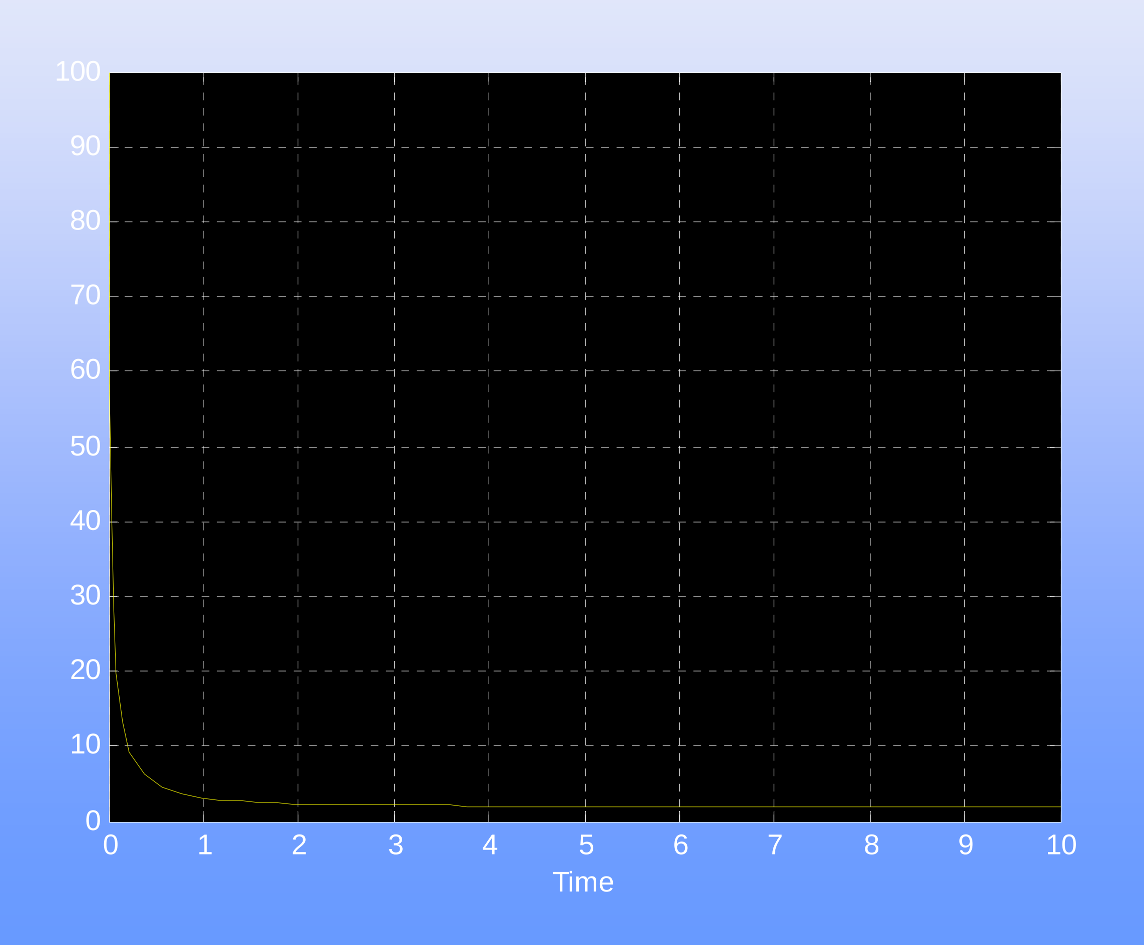


模型的起始时间默认为**0**，终止时间默认为**10.0**。需要改变终止时间时，旋转模型菜单【**Simulink>Parameters**】，打开模型参数对话框，设置【**Stop time**】为**10.0**。

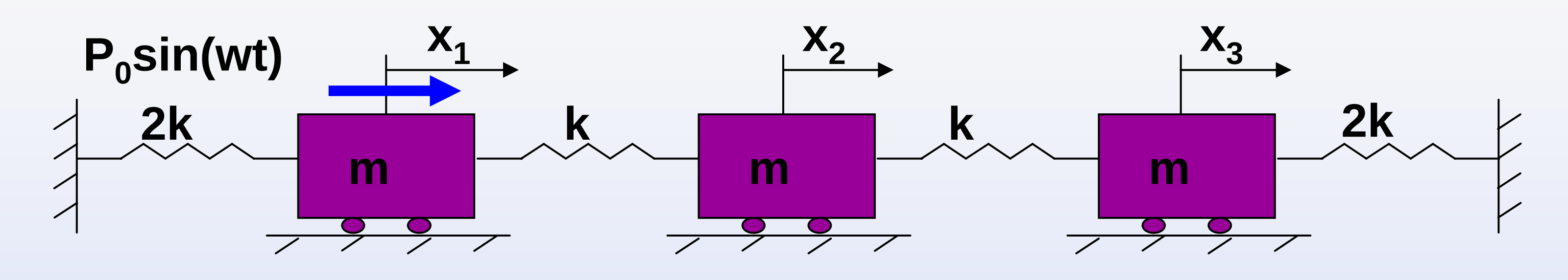
现在模型就全部完成了，选择【**File>Save**】命令保存模型为**bio_example**，**Simulink** 将以 **bio_example.mdl** 为文件名保存到指定的位置。

步骤四：运行模型

双击示波器模块，并选择【**Simulink>Start**】命令运行模型，示波器将绘制出仿真结果。



例子2：三自由度结构的强迫振动



动力学方程：

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix}\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{x}_3 \end{bmatrix} + \begin{bmatrix} 3k & -k & 0 \\ -k & 2k & -k \\ 0 & -k & 3k \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} P_0 \sin(\omega t) \\ 0 \\ 0 \end{bmatrix}$$

写成矩阵形式： $\boldsymbol{M}\ddot{\boldsymbol{X}} + \boldsymbol{K}\boldsymbol{X} = \overline{\boldsymbol{P}}(t)$

$$\overline{\boldsymbol{P}}(t) = \begin{bmatrix} P_0 \sin(\omega t) \\ 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \boldsymbol{M} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix} \quad \boldsymbol{K} = \begin{bmatrix} 3k & -k & 0 \\ -k & 2k & -k \\ 0 & -k & 3k \end{bmatrix}$$

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix} \begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{x}_3 \end{bmatrix} + \begin{bmatrix} 3k & -k & 0 \\ -k & 2k & -k \\ 0 & -k & 3k \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} P_0 \sin(\omega t) \\ 0 \\ 0 \end{bmatrix}$$

已知参数： m=1， k=1, P_0=1， $\omega = 0.5$

解析解：倪振华《振动力学》**P189 P233**

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} = -0.088 \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \frac{P_0}{k} \sin(wt) - 2.63 \begin{bmatrix} -\sqrt{3} \\ 0 \\ \sqrt{3} \end{bmatrix} \frac{P_0}{k} \sin(wt) + 0.21 \begin{bmatrix} \sqrt{2} \\ -\sqrt{2} \\ \sqrt{2} \end{bmatrix} \frac{P_0}{k} \sin(wt)$$

要求：

采用 **Simulink** 对系统进行动态仿真，并与解析解进行对照，计算时间 **0 ~ 50**

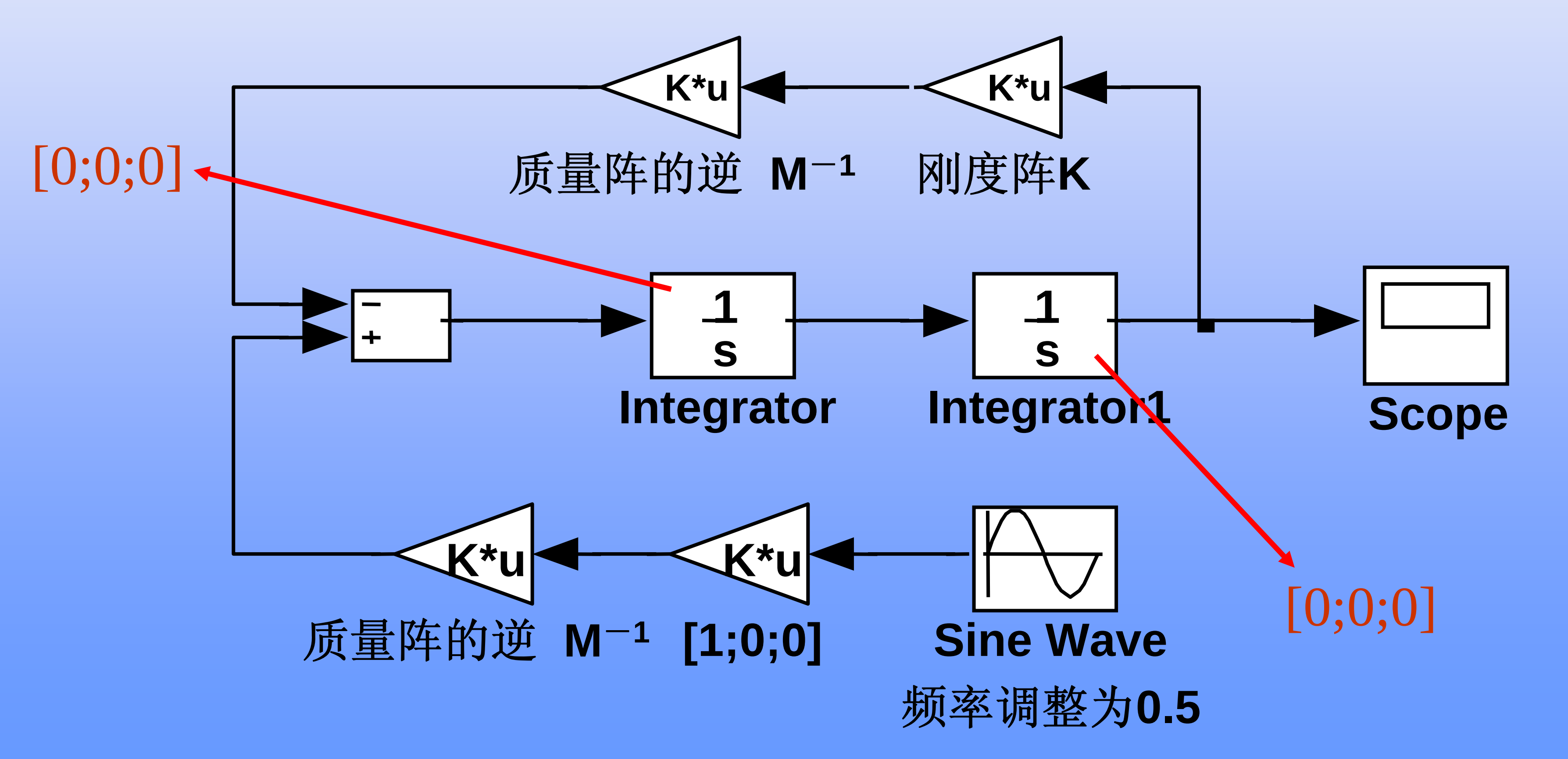
$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix}\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{x}_3 \end{bmatrix} + \begin{bmatrix} 3k & -k & 0 \\ -k & 2k & -k \\ 0 & -k & 3k \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} P_0 \sin(\omega t) \\ 0 \\ 0 \end{bmatrix}$$

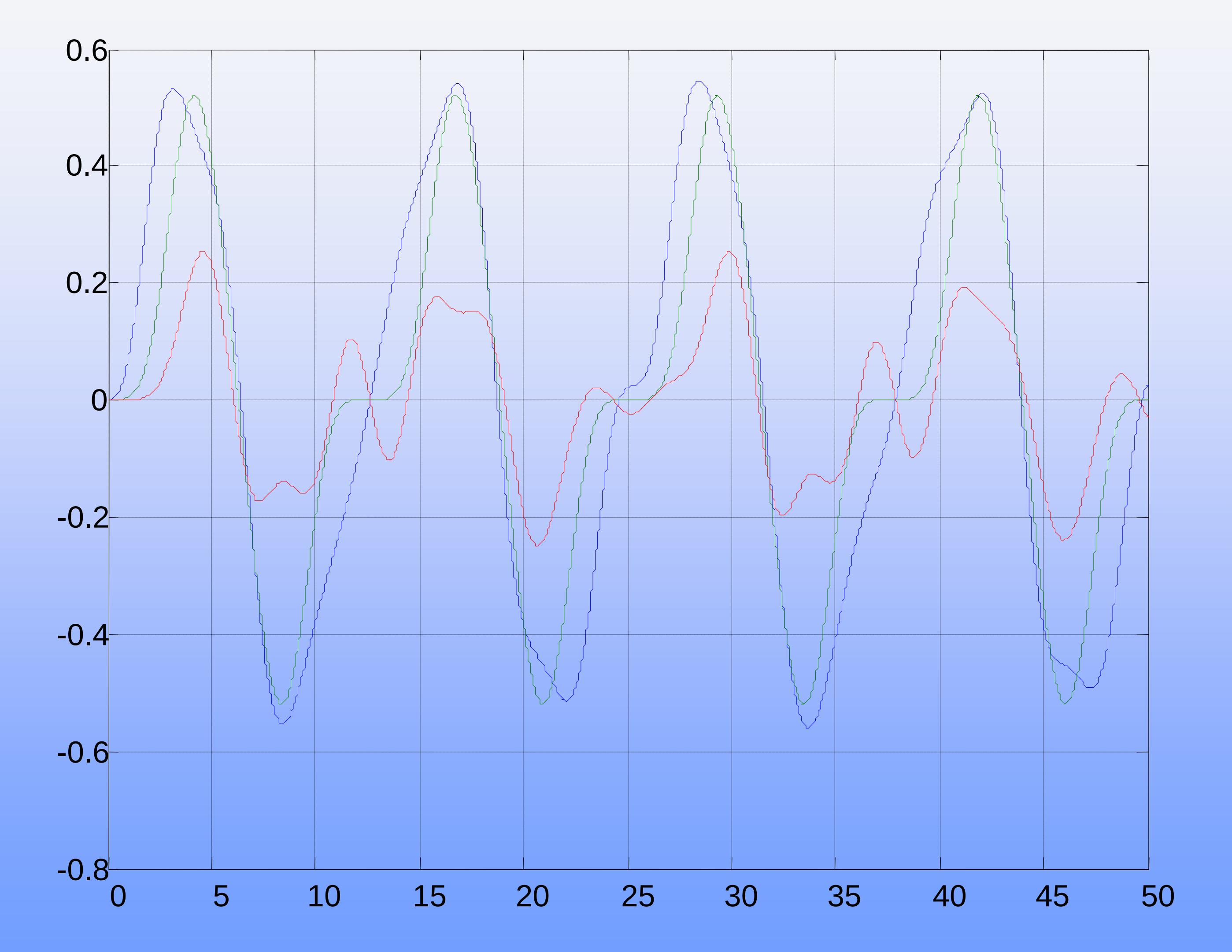
$$\boldsymbol{M}\ddot{\boldsymbol{X}} + \boldsymbol{KX} = \overline{\boldsymbol{P}}(t)$$

已知参数： m=1， k=1，

$$\ddot{\boldsymbol{X}} = -\boldsymbol{M}^{-1}\boldsymbol{KX} + \boldsymbol{M}^{-1}\overline{\boldsymbol{P}}(t)$$

P_0=1， $\omega = 0.5$





0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
5
10
15
20
25
30
35
40
45
50

2.6 模型和图形拷贝到Word文档中

2.6.1 拷贝到Word文档

使用【**Edit>Copy Model**】命令，然后在 **Word** 中粘贴

2.6.2 在Word文档中编辑模型和图形

使用取消组合命令

SIMULINK (3)

运行仿真

本章内容和学习目的

- 介绍两种 **Simulink** 运行仿真的方法
  - **3.1** 使用窗口运行仿真
  - **3.2** 使用 **MATLAB** 命令运行仿真
- 掌握以上两种运行仿真的方法

3.1 使用窗口运行仿真

优点：人机交互性强，不必记住繁琐的命令语句即可进行操作。使用窗口运行仿真主要可以完成以下一些操作。

1. 设置仿真参数	仿真参数和算法选择的设置
2. 应用仿真参数	仿真参数和算法设置后，使之生效
3. 启动仿真	选择命令运行仿真
4. 停止仿真	选择命令停止仿真
5. 中断仿真	可以在中断点继续启动仿真，而停止仿真则不能
6. 仿真诊断	在仿真中若出现错误，**Simulink** 将会终止仿真并在仿真诊断对话框中显示错误信息

1. 设置仿真参数

选择菜单选项【Simulation>Parameters】，可以对仿真参数及算法进行设置，共有五个选项卡

- 解法设置（Solver）
- 工作间I/O（Workspace I/O）
- 诊断页（Diagnostics）
- 高级设置（Advanced）
- 实时工具对话框（Real-Time Workshop）

- 解法设置（Solver）（讲）

设置起始和终止时间，选择积分分解法，指定求解参数和选择输出选项

- 工作间I/O（Workspace I/O）（讲）

管理MATLAB工作间的输入输出项

- 诊断页（Diagnostics）（不讲，自学）

选择在仿真中警告信息的等级

- 高级设置（Advanced）（不讲，自学）

对仿真的一些高级配置进行设置

- 实时工具对话框（Real-Time Workshop）（不讲，自学）

对实时工具中若干参数进行设置。若没有安装实时工具，不出现此框。

（1） 解法设置页

当选中菜单选项**【Simulation>Parameters】**后，出现参数及算法等设置页。再点击**【Solver】**，则出现解法设置页。解法设置页包括三项内容：

- 设置仿真的启动时间和终止时间
- 选择算法并指定参数
- 选择输出项

仿真时间

仿真解法（各种解法说明见下页）

默认解法（**ode45**）

变步长解法：**ode45，ode23，ode113，ode15，discrete**

定步长解法：**ode5，ode4，ode3，ode2，ode1，discrete**

最大步长

初始步长

输出选项：用户用来控制仿真输出个数的对话框，共有三个菜单选项：定义输出，产生附加输出，产生指定输出。

各种ode命令的说明

解法指令	解题类型	特　点	适合场合
ode45	非刚性	采用**4**、**5**阶**Runge－Kutta**法	大多数场合的首选算法
ode23	非刚性	采用**Adams**算法	较低精度（10^{-3}）场合
ode113	非刚性	多步法；采用**Adams**算法；高低精度均可（10^{-3}～10^{-6}）	**ode45**计算时间太长时取代**ode45**
ode23t	适度刚性	采用梯形法则算法	适度刚性
ode15s	刚性	多步法；采用**2**阶**Rosenbrock**算式，精度中等	当**ode45**失败时使用；或存在质量矩阵时
ode23s	刚性	一步法；采用**2**阶**Rosenbrock**算式，低精度	低精度时，比**ode15s**有效；或存在质量矩阵时
ode23tb	刚性	采用梯形法则一反向数值微分两阶段算法，低精度	低精度时比**ode15s**有效；或存在质量矩阵时

（2） 工作间 I/O

Simulink 作为 **MATLAB** 的一个附件，理应与 **MATLAB** 很好地结合，它的输入数据可以从 **MATLAB** 的工作空间中获得，其仿真结果也可以被引入到 **MATLAB**的 工作空间。实现此项功能需要用到仿真参数对话框中的工作间 **I/O** 页。

工作间 **I/O** 页大致可以分为三个部分：

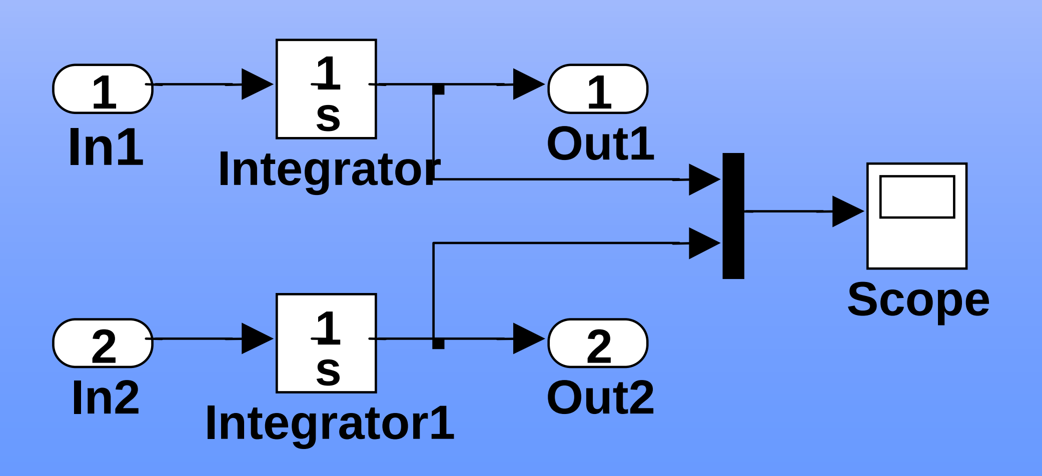
- 从**MATLAB**工作空间获得系统输入（**Load from workspace**）
- 仿真结果输出到**MATLAB**的工作空间（**Save to workspace**）
- 输出选项（**Save option**）

下面对 **Workspace I/O** 选项页的功能与使用分别予以简介。

◆ 从MATLAB工作空间加载（Load from workspace）

虽然 Simulink 提供了多种系统输入信号，但并不能完全满足需要。Simulink允许使用用户自定义的信号作为系统输入信号。在 Load from workspace 框中，用户可以设置 MATLAB 中的变量作为系统输入信号或系统状态初值，如下所述：

（1） Input：用来设置系统输入信号，其格式为 [t, u]，其中 t、u 均为列向量，t 为输入信号的时间向量，u 为相应时刻的信号取值。可以使用多个信号输入，如 [t, u1, u2]。输入信号与 Simulink 的接口由 Inport 模块（In1模块）实现。



◆ 从MATLAB工作空间加载（Load from workspace）

虽然 Simulink 提供了多种系统输入信号，但并不能完全满足需要。Simulink允许使用用户自定义的信号作为系统输入信号。在 Load from workspace 框中，用户可以设置 MATLAB 中的变量作为系统输入信号或系统状态初值，如下所述：

（1） Input：用来设置系统输入信号，其格式为 [t, u]，其中 t、u 均为列向量，t 为输入信号的时间向量，u 为相应时刻的信号取值。可以使用多个信号输入，如 [t, u1, u2]。输入信号与 Simulink 的接口由 Inport 模块（In1模块）实现。

（2） Initial state：用来设置系统状态变量的初始值。初始值 xInitial 可为列向量。

注意：使用 xInitial state 所设置的状态变量初始值会自动覆盖系统模块中的设置。另外，输入信号与状态变量需要按照系统模型中 Inport 模块（即 In1 模块）的顺序进行正确设置。

◆ 仿真结果输出到MATLAB的工作空间

（Save to workspace）

使用 **Workspace I/O** 选项页可以将系统的仿真结果、系统仿真时刻、系统中的状态或指定的信号输出到 **MATLAB** 的工作空间中，以便用户对其进行定量分析，如下所述：

（1） **Time**：输出系统仿真时刻。

（2） **States**：输出系统模型中的所有状态变量。

（3）**Output**：输出系统模型中的所有由 **Output** 模块（即 **Out1** 模块）表示的信号。

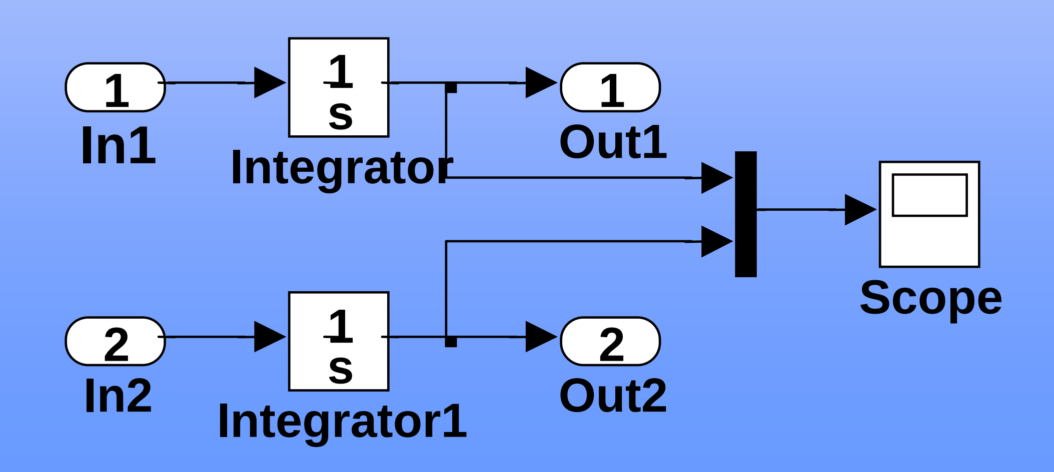
（4）**Final state**：输出系统模型中的最终状态变量取值，即最后仿真时刻处的状态值。

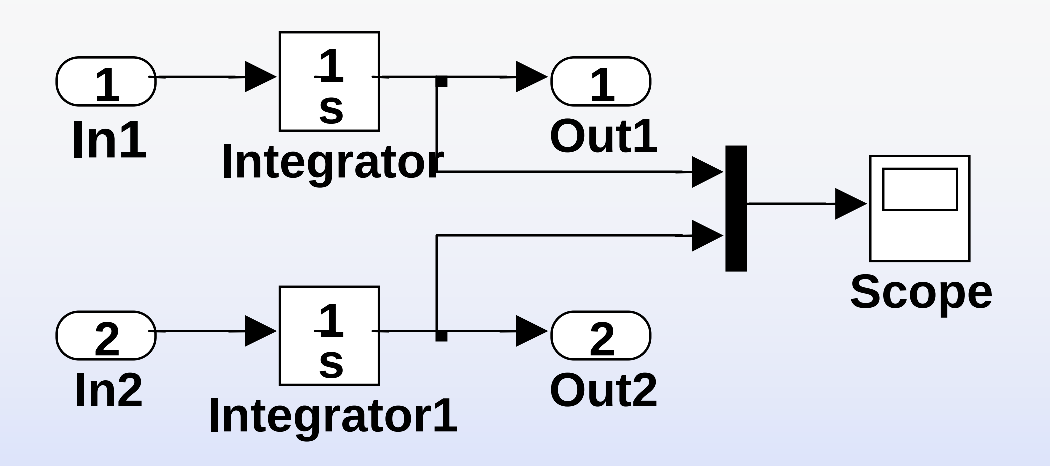
◆ 输出选项（Save option）

（1） **Limit data points to last**：表示输出数据的长度（从信号的最后数据点记起）。

（2） **Format**：表示输出数据类型。共有三种形式：**Structure with Time**（带有仿真时间变量的结构体）、**Structure**（不带仿真时间变量的结构体）、**Array**（信号数组）。

举例说明:

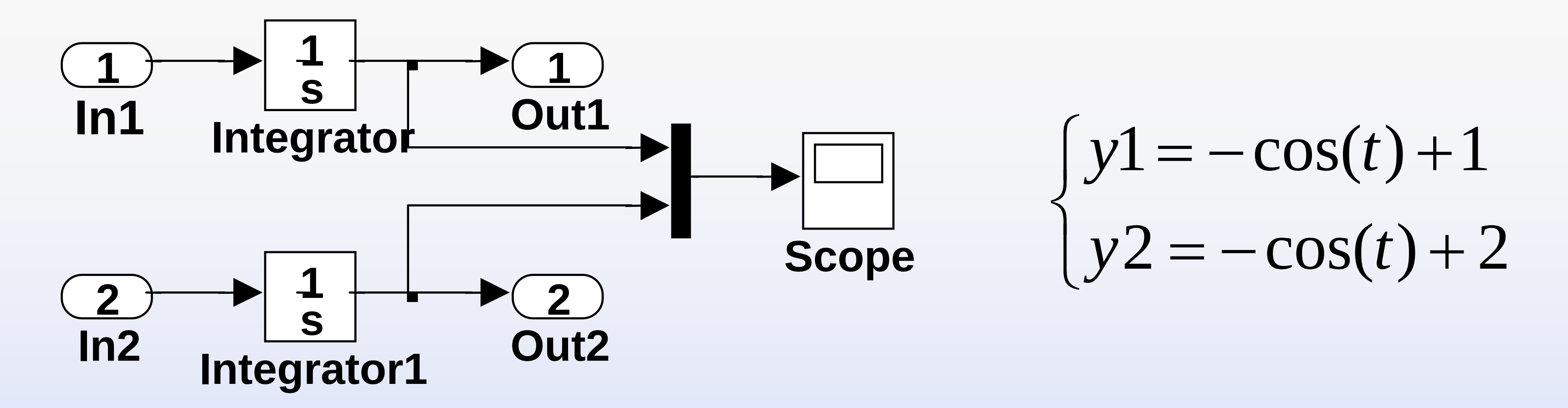




假定两个输入信号都为正弦信号 **sin(t)**，并且假定初始值为 **[0，1]**

$$y = \int \sin(t) = -\cos(t) + C$$

$$\begin{cases} y1 = -\cos(t) + 1 \\ y2 = -\cos(t) + 2 \end{cases}$$



将 **Load from workspace** 栏的两项全部选上，并且将 **Input** 栏改写为**[t, u, u]**。

将 **Save to workspace** 栏的四项全部选上。

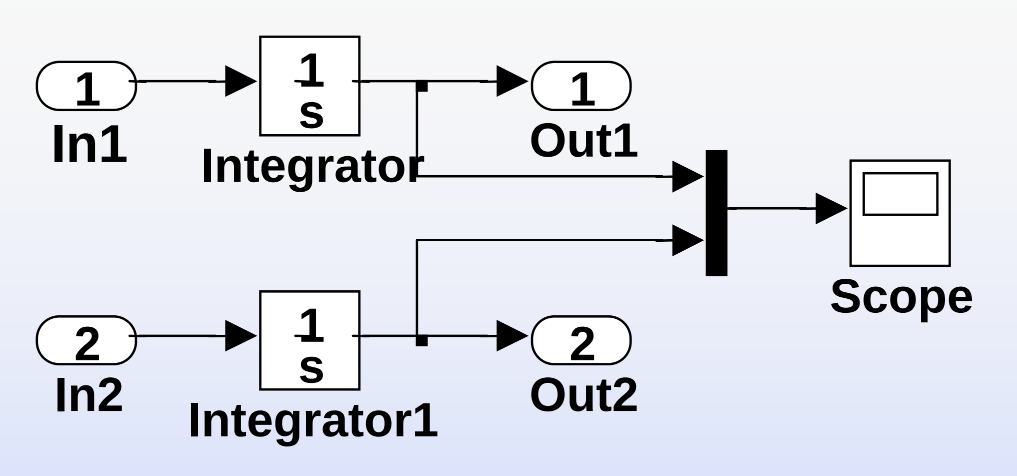
Save option 栏的三项分别为：**1000**，**1**，**Array**

运行仿真前，首先需要生成系统输入信号与状态初始值，在 **MATLAB** 命令窗口中键入如下命令：

```
> t = 0:  0.1:  10;   t = t’;

> u = sin ( t );

> xInitial = [0,  1];
```



$$\begin{cases} y1 = -\cos(t) + 1 \\ y2 = -\cos(t) + 2 \end{cases}$$

运行 **whos** 的结果：

Name	Size	Bytes	Class
t	101x1	808	double array
tout	51x1	408	double array
u	101x1	808	double array
xFinal	1x2	16	double array
xInitial	1x2	16	double array
xout	51x2	816	double array
yout	51x2	816	double array

上面 **t** 和 **tout** 的维数不相同，这是因为在 **Solver** 中采用了变步长解法，若采用定步长解法（步长取 **0.1**）则维数相同。

3.2 使用MATLAB命令运行仿真

MATLAB 提供了 **sim** 命令，用户可以在 **MATLAB** 的环境下以命令行或 **M** 文件的形式运行 **Simulink** 模型。

使用命令行方式，用户可以在脚本文件中重复地对同一系统在不同的仿真参数或不同的系统模块参数下进行仿真，而无需一次又一次启动 **Simulink** 图形窗口中的 **Start Simulink**。

如果需要分析某一参数对系统仿真结果的影响，用户可以很容易地通过 **for** 循环自动修改任意指定的参数。

下面对进行动态系统仿真的命令逐一介绍。首先看一个例子。

建立一个简单的动态系统，其功能如下：

（1）系统的输入为一单位幅值、单位频率的正弦信号；

（2）系统的输出信号为输入信号的积分。

$$\sin(t) \qquad\qquad \int \sin(t)dt$$

要求如下：

（1）系统的输入信号由 **MATLAB** 工作空间中的变量提供，时间 **0～10s**；

（2）使用 **MATLAB** 绘制原始输入信号与系统运算结果的曲线。

问题描述：

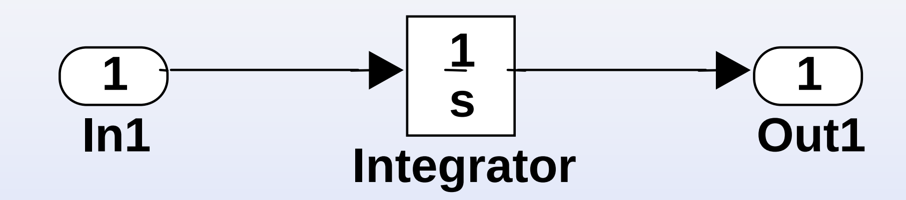
原函数： $\sin(t)$　　时间区域： $t = 0 \sim 10s$

原函数的积分为： $x(t) = \int \sin(t)dt = -\cos(t) + C$

若取零初始条件，则有： $C = 1$

因此，有： $x(t) = -\cos(t) + 1$

$$x(t) = -\cos(t) + 1$$



Simulink 模型

（保存文件名为：**command_in_out**）

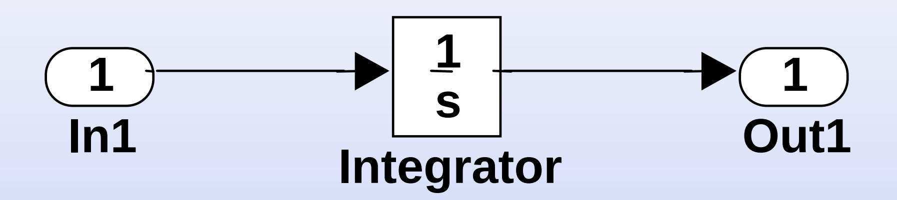
对 **Workspace I/O** 页进行如下设置：

（**1**）**Load from workspace** 栏：**Input** 打勾，并填入 **sim_input**（由 **MATLAB** 工作空间输入的输入名）。**Initial state** 不选。

（**2**）**Save to workspace** 栏：**Time** 和 **Output** 项勾上，**State** 和 **Final state** 项不勾。

（**3**）**Save options** 栏依次为：**1000**，**1**，**Array**

$$x(t) = -\cos(t) + 1$$



在 **MATLAB** 工作空间中定义输入变量 **sim_input** 如下：

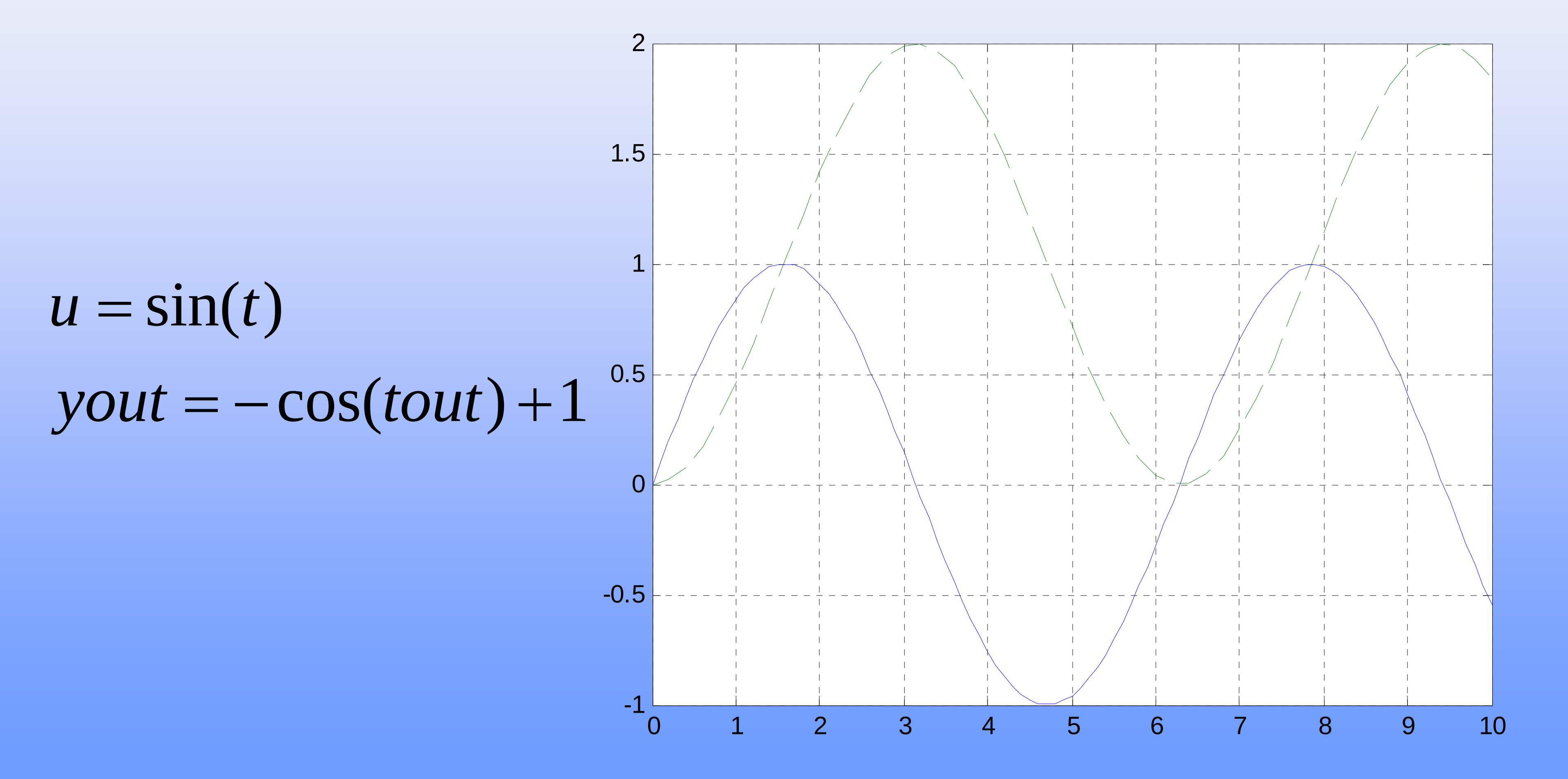
>t = 0: 0.1: 10; t = t'; % 表示输入信号的时间范围

>u = sin (t); % 产生输入正弦信号

>sim_input = [t, u]; % 传递给 **Simulink** 系统模型的变量

然后运行 **Simulink** 仿真。最后在 **MATLAB** 工作空间运行如下命令：

```
plot (t, u, tout, yout, '--');   grid
```



3.2.1 使用 sim 命令进行动态系统仿真

1 使用语法

sim 命令的格式为：

[t, x, y] = sim (model, timespan, options, ut)

[t, x, y1, y2, …, yn] = sim (model, timespan, options, ut)

以上是完整的语法格式，实际使用时可以省略其中的某些参数设置而采用默认参数。除了参数 “model” 外，其它的仿真参数设置均可以取值为空矩阵，此时 sim 命令对没有设置的仿真参数使用默认的参数值进行仿真，默认的参数值由系统模型框图决定。用户可以使用 sim 命令的 options 参数对可选参数进行设置，这样设置的仿真参数将覆盖模型默认的参数。

2 参数说明

sim 命令的格式为：

[t, x, y] = sim (model, timespan, options, ut)

[t, x, y1, y2, …, yn] = sim (model, timespan, options, ut)

（1）model：需要进行仿真的系统模型框图名称；

（2）timespan：系统仿真的时间范围（起始至终止时间），可有如下形式：

tFinal：设置仿真终止时间。仿真起始时间默认为0；

[tStart tFinal]：设置起始时间 (tStart)与终止时间（tFinal）；

[tStart OutputTimes tFinal]：设置起始时间（tStart）与终止时间（tFinal），并且设置仿真返回的时间向量 [tStart OutputTimes tFinal]，其中 tStart、OutputTimes、tFinal 必须按照升序排列。

sim 命令的格式为：

[t, x, y] = sim (model, timespan, options, ut)

[t, x, y1, y2, …, yn] = sim (model, timespan, options, ut)

（3）options：由 simset 命令所设置的除仿真时间外的仿真参数；

（4）ut：表示系统模型顶层的外部可选输入。ut 可以是 MATLAB 函数。可以使用多个外部输入 ut1、ut2、…。

（5）t：返回系统仿真的时间向量。

（6）x：返回系统仿真的状态变量矩阵。

（7）y：返回系统仿真的输出矩阵。按照顶层输出 Outport 模块的顺序输出。如果输出信号为向量输出，则输出信号具有与此向量相同的维数。

（8）y1，…，yn：返回多个系统仿真的输出。

3 举例之一：简单仿真

输入：sin(*t*)

$$x(t) = -\cos(t) + 1$$

文件名： **command_in_out**

对于前面的动态系统 **command_in_out**，在前面进行仿真时没有使用命令行方式，在此采用命令行语句进行仿真。在仿真之前，首先使用仿真参数设置对话框设置参数，然后在 **MATLAB** 命令窗口中键入如下命令：

```
>t = 0: 0.1: 10;  t = t';      % 表示输入信号的时间范围
>u = sin ( t );                % 产生输入正弦信号
>sim_input = [ t, u ];         % 传递给 Simulink 系统模型的变量
>[ tout, x, yout ]  = sim ( 'command_in_out' )
                  % 使用 sim 进行系统仿真，仿真参数取与前面相同
>plot ( t, u, tout, yout, '--' );   grid
```

4 举例之二：仿真时间设置

在前面已经对 **sim** 命令中的仿真时间参数 **timespan** 设置做了介绍。 **Timespan** 具有三种使用形式，根据不同动态系统的不同要求，用户可以选择使用如下所示的三种形式进行系统仿真：

```
[ t, x, y ]  = sim ( model, tFinal )

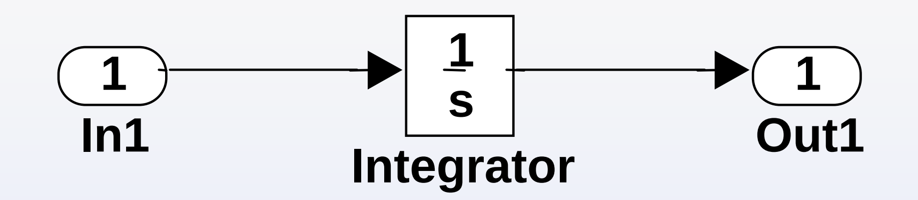
[ t, x, y ]  = sim ( model, [ tStart,  tFinal] )

[ t, x, y ]  = sim ( model, [ tStart  outputTimes  tFinal ] )
```

```
[ t, x, y ]  = sim ( model, tFinal )
[ t, x, y ]  = sim ( model, [ tStart,  tFinal] )
[ t, x, y ]  = sim ( model, [ tStart  outputTimes  tFinal ] )
```

此外在默认情况下，系统仿真的输出结果（输出时间、状态和运算结果）受到 **Simulink** 求解器仿真步长的控制，因而系统仿真输出结果也受到求解器步长的控制。

如果需要在指定的时刻输出系统仿真结果，则需要使用仿真时间设置的第三种方式，其中 **[tStart outputTimes tFinal]** 表示输出时间向量，此向量为一递增行向量。在 **sim** 命令中使用 **timespan** 设置系统仿真时间范围，会覆盖 **Simulink** 原来的仿真时间设置，但是并不会影响到系统模型。



输入：**sin(*t*)**　　　　$x(t) = -\cos(t) + 1$

仍以前面的 **command_in_out** 为例进行说明。仿真参数设置对话框内的设置与前面相同。使用四组不同的仿真时间对此系统进行仿真。

为了与系统模型文件 **command_in_out.mdl** 区别，所编制的 **M** 文件名取为 **command_in_out_m_m**。若 **M** 文件名取为 **command_in_out.m**，则在 **MATLAB** 工作空间运行时，将返回系统模型文件框图。

M文件名：command_in_out_m

```
>t=0:0.1:10;   t=t';      u=sin(t);      sim_input=[t,u];

>[tout1,x1,yout1]=sim('command_in_out',5);  % 仿真时间范围0～5s，输出时间向量 tout1 由 Simulimk 的求解器步长决定

>[tout2,x2,yout2]=sim('command_in_out',[1  8]);  % 仿真时间范围1～8s，输出时间向量 tout2 由 Simulimk 的求解器步长决定

>[tout3,x3,yout3]=sim('command_in_out',[1:8]);  %  仿真时间范围1～8s，并且每隔 1s 输出一次，即输出时间变量为 [1 2 3 4 5 6 7 8]

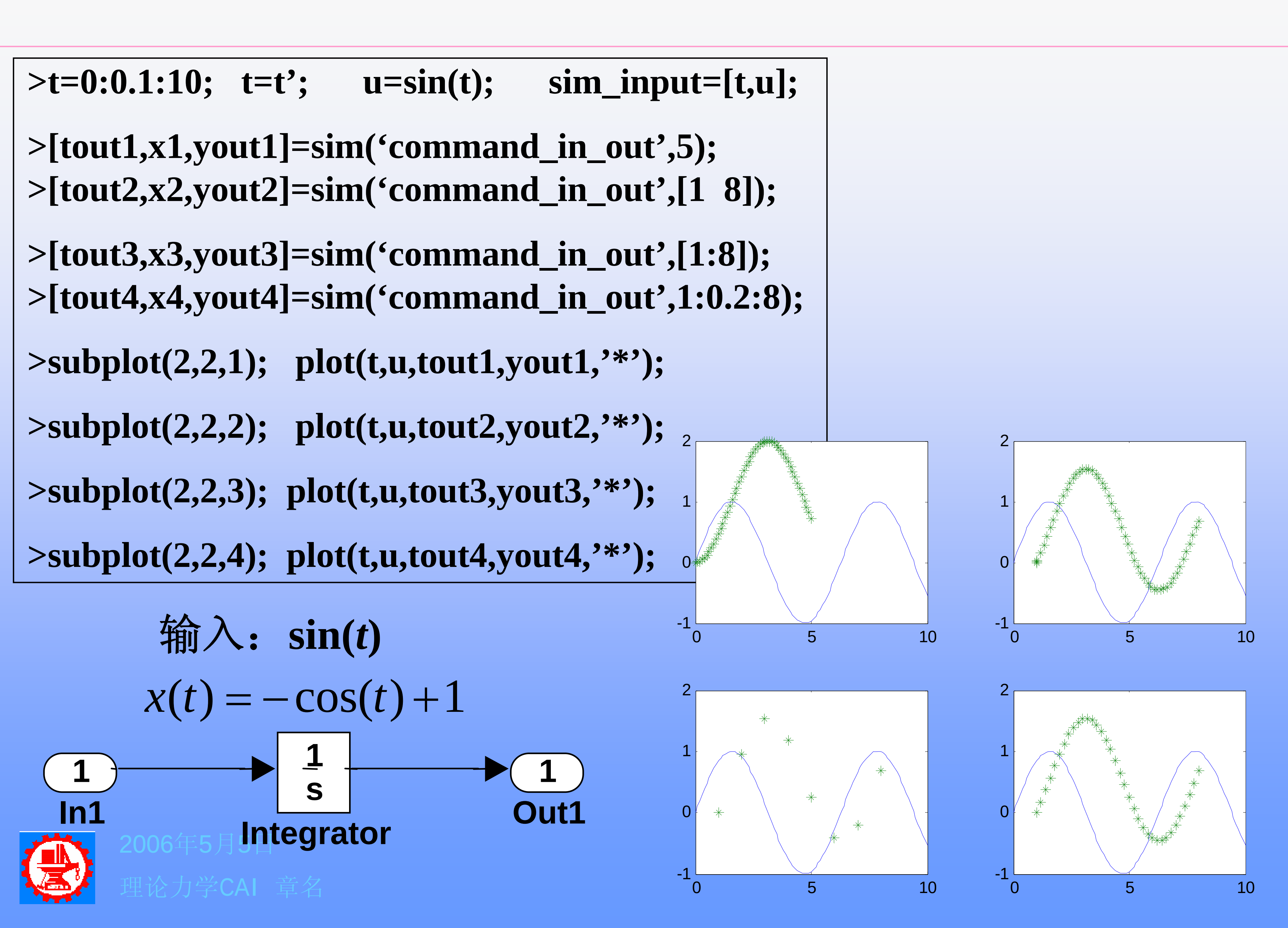
>[tout4,x4,yout4]=sim('command_in_out',1:0.2:8);  %  仿真时间范围1～8s，并且每隔 0.2s 输出一次，即输出时间变量为 [1 1.2 1.4  …  7.6  7.8  8]

>subplot(2,2,1);   plot(t,u,tout1,yout1,'*');

>subplot(2,2,2);   plot(t,u,tout2,yout2,'*');

>subplot(2,2,3);  plot(t,u,tout3,yout3,'*');

>subplot(2,2,4);  plot(t,u,tout4,yout4,'*');
```



>t=0:0.1:10; t=t’; u=sin(t); sim_input=[t,u];
>[tout1,x1,yout1]=sim(‘command_in_out’,5);
>[tout2,x2,yout2]=sim(‘command_in_out’,[1 8]);
>[tout3,x3,yout3]=sim(‘command_in_out’,[1:8]);
>[tout4,x4,yout4]=sim(‘command_in_out’,1:0.2:8);
>subplot(2,2,1); plot(t,u,tout1,yout1,’*’);
>subplot(2,2,2); plot(t,u,tout2,yout2,’*’);
>subplot(2,2,3); plot(t,u,tout3,yout3,’*’);
>subplot(2,2,4); plot(t,u,tout4,yout4,’*’);
输入：sin(t)
x(t) = −cos(t) + 1
1
In1
1
s
Integrator
1
Out1
2006年5月5日
理论力学CAI 章名

5 举例之三：外部输入变量设置

前面对动态系统 **command_in_out** 进行仿真时，通过设置 **Simulink** 仿真参数设置对话框中 **Workspace I/O** 中的外部变量输入，以使系统在仿真过程中获取输入信号 **sim_input**。除了使用这种方法从 **MATLAB** 工作空间中获得系统输入信号之外，用户还可以通过使用 **sim** 命令中的 **ut** 参数来设置系统的外部输入信号。下面介绍如何使用 **ut** 参数设置外部输入信号。

仍以 **command_in_out** 为例进行说明。使用如下命令：

[t, x, y]=sim (model, timespan, options, ut)

ut 为一个具有两列的矩阵，第一列表示外部输入信号的时间，第二列代表与时间列相对应的外部输入信号。

输入：**sin(*t*)**

$$x(t) = -\cos(t) + 1$$

1
In1
1/s
Integrator
1
Out1

MATLAB 程序（程序名：command_in_out_m2.m）

>t=0:0.1:10;　t=t’;　u=sin(t);　sim_input=[t, u];

>[tout1, x1, yout1]=sim(‘command_in_out’, 10);　% 使用 Simulink 仿真参数对话框中的 workspace I/O 从 MATLAB 工作空间中获得输入信号

>u=cos(t);

>ut=[t, u];　% 改变系统输入信号

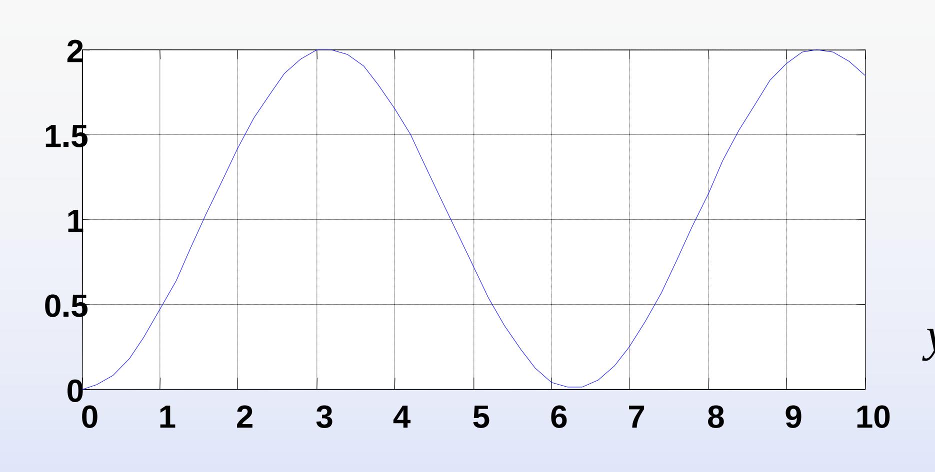
>>[tout2, x2, yout2]=sim(‘command_in_out’, 10, [], ut);　% 使用 Sim 中的 ut 参数获得系统输入信号， ut的使用会覆盖由 Workspace I/O 的系统输入设置，这一点可以在下面的系统仿真结果图形中反映出来

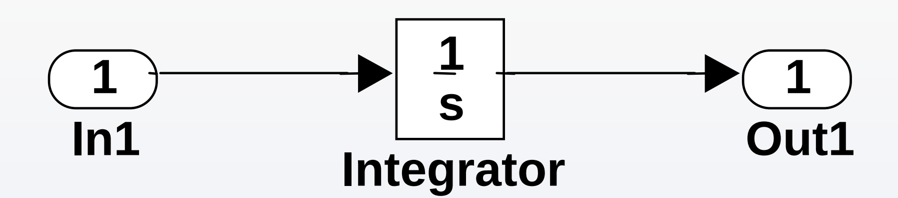
>subplot(1, 2, 1);　plot(tout1, yout1);　grid

>subplot(1, 2, 2);　plot(tout2, yout2);　grid

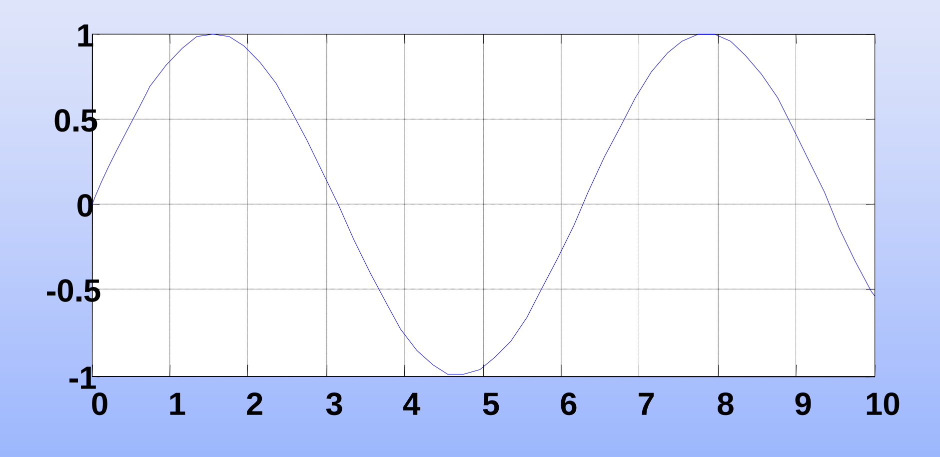
$$yout1=\int \sin(tout1)d(tout1)=-\cos(tout1)+1$$

$$yout2=\int \cos(tout2)d(tout2)=\sin(tout2)$$





$$yout1 = \int \sin(tout1)d(tout1) = -\cos(tout1) + 1$$



$$yout2 = \int \cos(tout2)d(tout2) = \sin(tout2)$$

图中上图表示系统输入为 **sin(t)** 时的相应曲线，下图表示 **cos(t)** 时的相应曲线。从图中可明显看出，当使用 **sim** 命令的 **ut** 参数时，**Simulink** 仿真参数设置对话框中的设置被覆盖。以前对话框中的外部输入是名为 **sim_input** 的正弦信号，而采用 **ut** 参数后执行的余弦输入信号。注意：这里指的“覆盖”，并不是在 **Workspace I/O** 对话框的 **Input** 中，将 **sim_input** 改变成了 **ut**，事实上并没有改变，只是不执行 **sim_input**，而执行了命令行中的 **ut**。

3.2.2 simset 命令

simset 命令是用来创建和编辑 **options** 结构的。

3.2.3 simget 命令

simget 命令用来获得系统模型的仿真参数设置。

3.2.4 simplot 命令

使用语法：

simplot (data);

simplot (time, data);

说明：**data** 是动态系统方针结果的输出数据，一般由 **outport** 模块、**To Workspace** 模块等产生的输出，其数据类型可以为矩阵、向量或是结构体等。

SIMULINK (4)

基本模块介绍

本章内容和学习目的

- **Simulink** 模块的基本知识
- 基本模块的性质分类和基本操作

4.1 连续系统模块

大多数物理系统可以用微分方程进行描述，因此可以用连续系统模拟。最简单的模型是线性模型和定常模型。

例如，振动理论中的动力学方程：

$$\boldsymbol{M}\ddot{\boldsymbol{x}}(t)+\boldsymbol{C}\dot{\boldsymbol{x}}(t)+\boldsymbol{K}\boldsymbol{x}(t)=\boldsymbol{P}(t)$$

其中，$\boldsymbol{x}$ 为系统的广义坐标列向量，$\boldsymbol{M}$ 为质量矩阵，$\boldsymbol{C}$ 为阻尼矩阵，$\boldsymbol{K}$ 为刚度矩阵，$\boldsymbol{P}(t)$为外部激励列向量。

在 **Simulink**中，用来模拟连续系统的基本模块有四个：增义模块，求和模块，微分模块，积分模块。除了这四个基本模块，传递函数模块也经常用来模拟物理系统和控制器。

1. 增益模块

作用：使增益模块的输入信号乘以一个常数，并输出。可用代数表达式表示为：

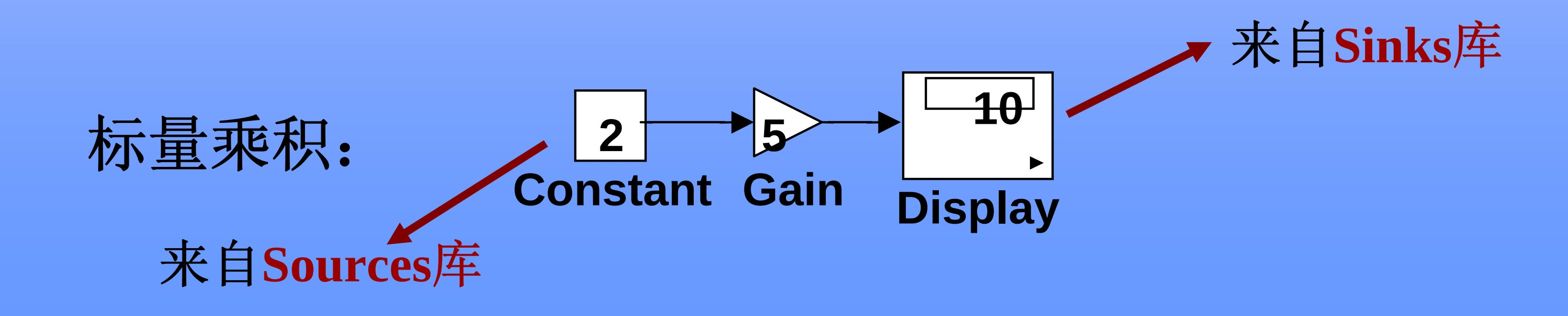
$$y(t) = k\,x(t)$$

简图如下：



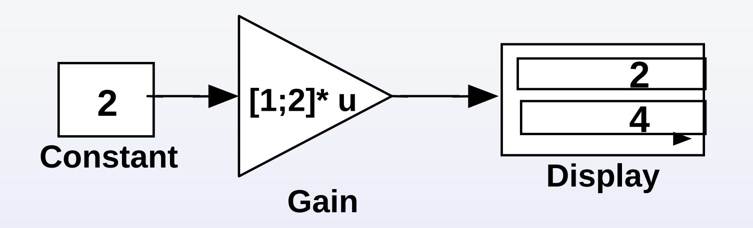
注意：$y(t)$、$x(t)$、k 可以为标量、向量或矩阵。

标量乘积：



标量和向量的乘积

$$2\times\begin{bmatrix}1\\2\end{bmatrix}=\begin{bmatrix}2\\4\end{bmatrix}$$



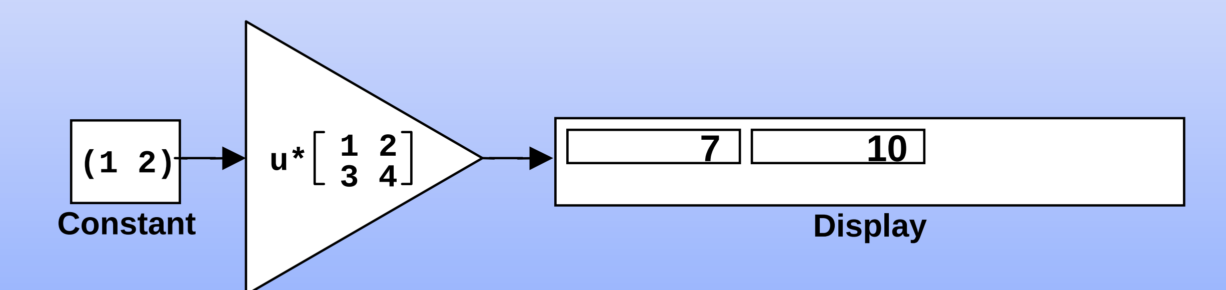
向量和标量的乘积

$$\begin{bmatrix}1 & 2\end{bmatrix}\times 2=\begin{bmatrix}2 & 4\end{bmatrix}$$

(1 2)
Constant
u *2
Gain
2 4
Display

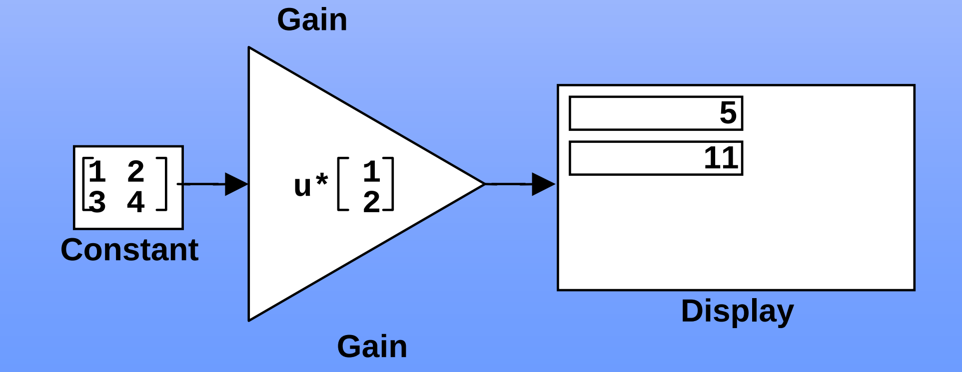
向量和矩阵的乘积

$$\begin{bmatrix}1 & 2\end{bmatrix}\times\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}=\begin{bmatrix}7 & 10\end{bmatrix}$$



矩阵和向量的乘积

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\times\begin{bmatrix}1\\2\end{bmatrix}=\begin{bmatrix}5\\11\end{bmatrix}$$



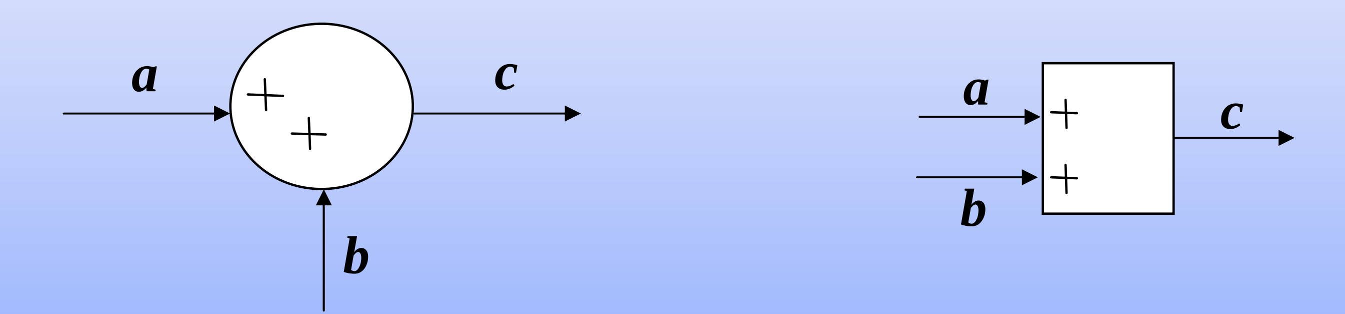
2. 求和模块

作用：对两个或多个信号进行求和运算。

可用代数表达式表示为：

$$c = a + b$$

两种形状：圆形和方形。

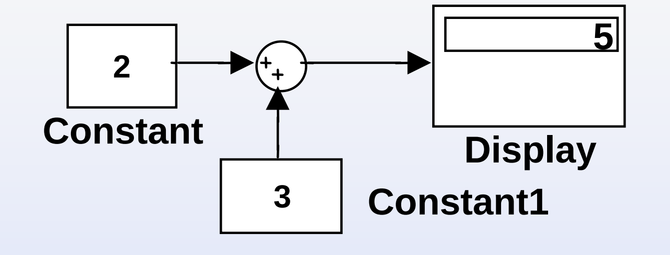


求和模块必须至少有一个输入而仅有一个输出。输入的正负号的数目由双击模块进入编辑栏进行设定。

求和模块不但可以进行标量求和运算，也可以进行向量或矩阵求和运算，但是标量或矩阵的维数必须相等。

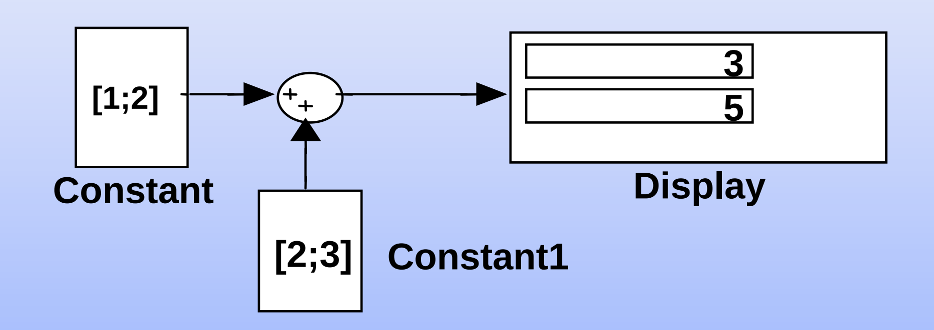
标量求和

$$2+3=5$$



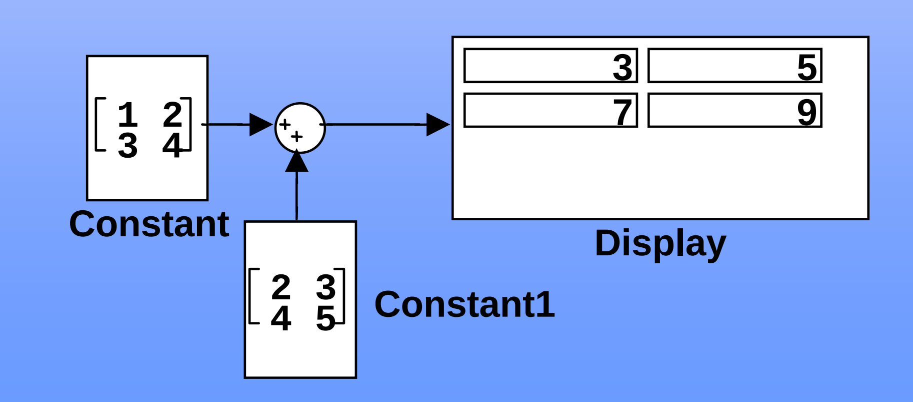
向量求和

$$\begin{bmatrix}1\\2\end{bmatrix}+\begin{bmatrix}2\\3\end{bmatrix}=\begin{bmatrix}3\\5\end{bmatrix}$$



矩阵求和

$$\begin{bmatrix}1&2\\3&4\end{bmatrix}+\begin{bmatrix}2&3\\4&5\end{bmatrix}=\begin{bmatrix}3&5\\7&9\end{bmatrix}$$

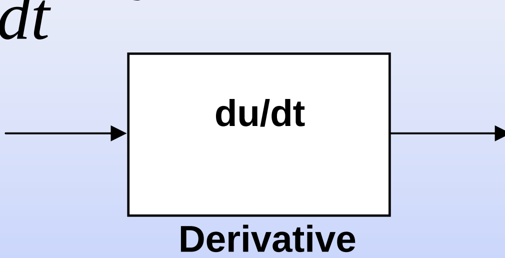


3. 微分模块

作用：计算输入对时间的变化率。

代表如下微分方程：
$$y=\frac{dx}{dt}=\dot{y}$$

微分模块如图所示：

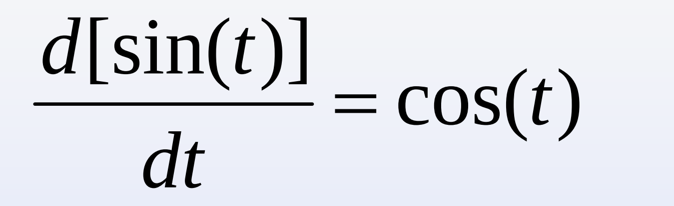


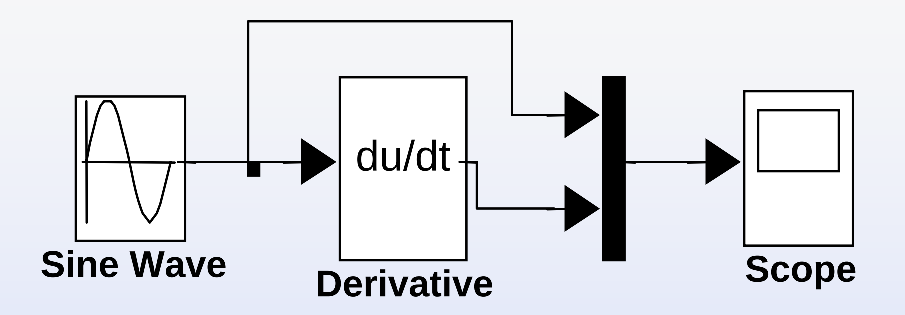
例如，对于动力学方程：

$$\boldsymbol{M}\ddot{\boldsymbol{x}}(t)+\underline{\boldsymbol{C}\dot{\boldsymbol{x}}(t)}+\boldsymbol{K}\boldsymbol{x}(t)=\boldsymbol{P}(t)$$

考虑对正弦信号 **sin (t)** 的微分：
$$\frac{d[\sin(t)]}{dt}=\cos(t)$$

Simulink 模型框图和仿真结果如下页图形所示。





1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

0 1 2 3 4 5 6 7 8 9 10

4. 积分模块

作用：计算输入信号从起始时间到当前时刻对时间的积分。

代表如下微分方程： $y(t)=y(t_0)+\int_{t_0}^{t}x(\tau)d\tau$

积分模块如图所示：

$\frac{1}{s}$

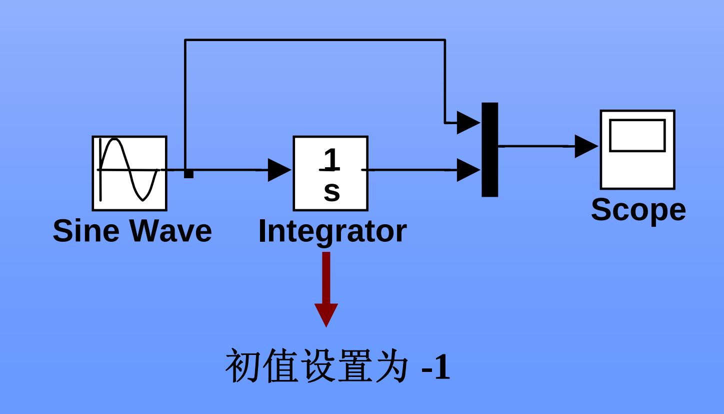
Integrator

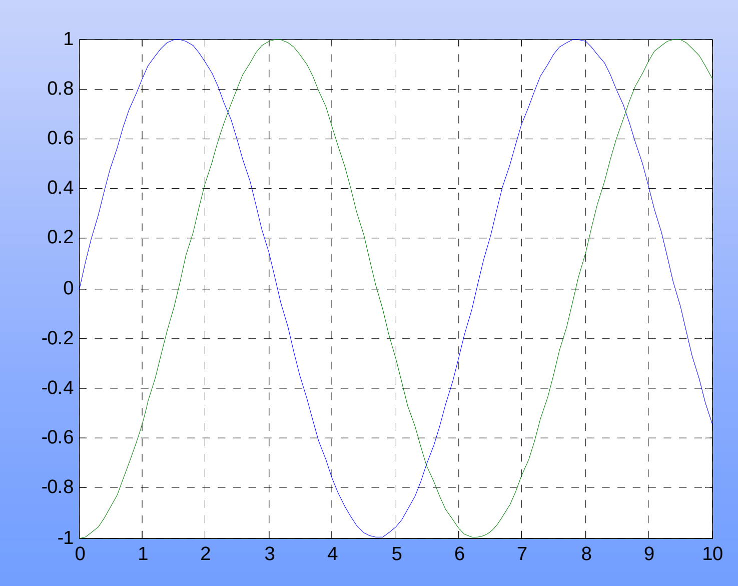
例如：

$$y(t)=\int\sin(t)dt$$

$$=-\cos(t)+C=-\cos(t)$$

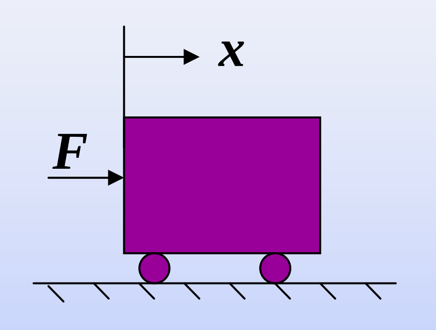
注：$y(t)$ 的初值假设为-1





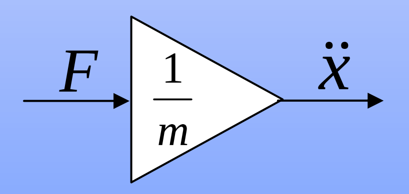
5. 简单物理模型

利用前面所介绍的这些模块可以模拟由线性微分工程描述的任何物理模型。例如，考虑如下所示的简单的小车系统运动。

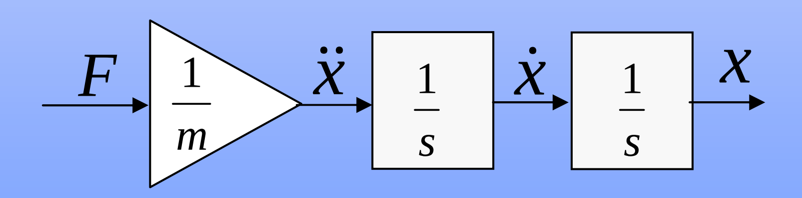


忽略摩擦力，运动微分方程为：
$$m\ddot{x} = F \qquad \ddot{x} = \frac{F}{m}$$

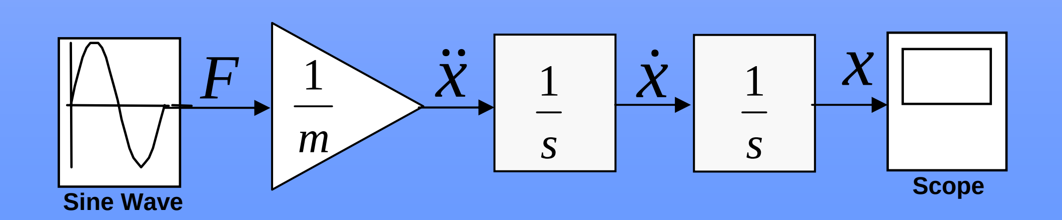
可用模块图表示为：



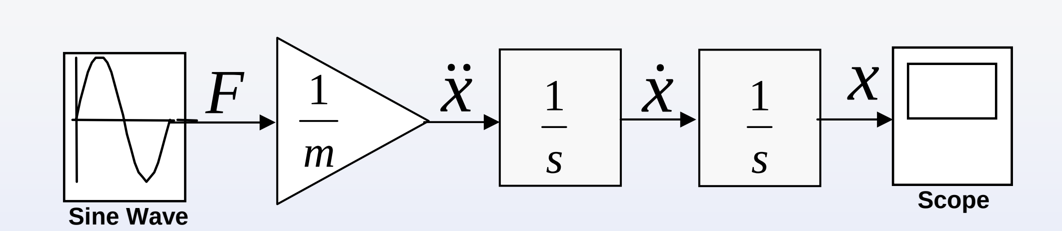
加入两个积分模块，第一个模块用来计算速度，第二个模块用来计算位移：

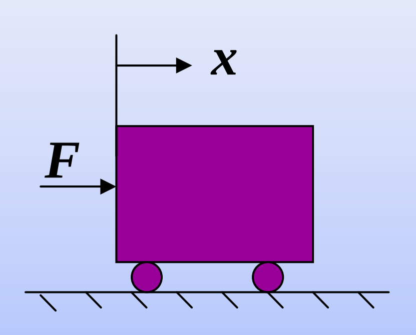


假定 F=sin(t) 为正弦激励，m=0.5。求0～20s 区间内的系统位移响应曲线。

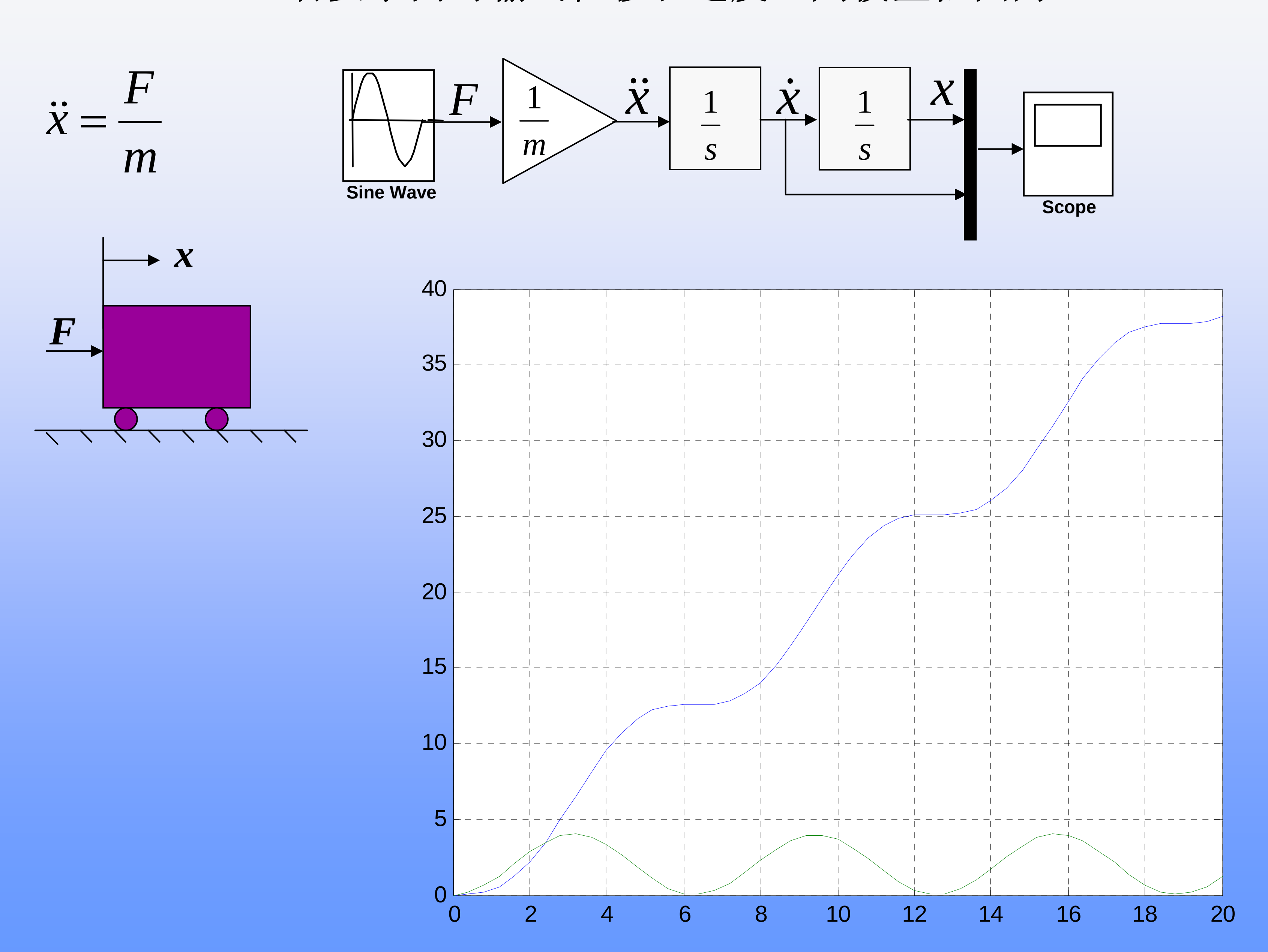


$$\ddot{x} = \frac{F}{m}$$

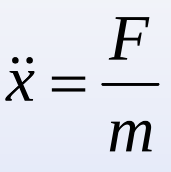


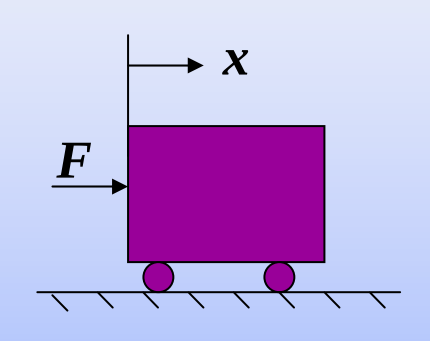


若要求同时输出位移和速度，则模型框图为：



若要求同时输出位移、速度和加速度，则模型框图为：





Sine Wave

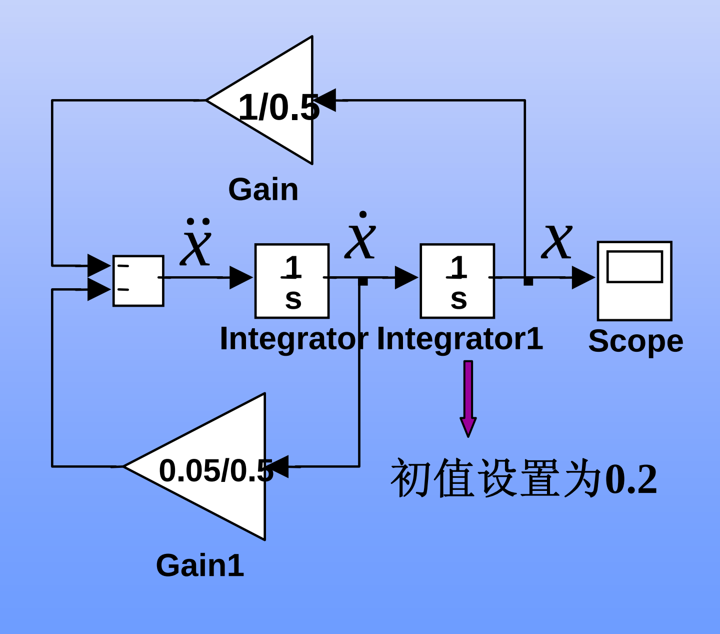
F → $\frac{1}{m}$ → $\ddot{x}$ → $\frac{1}{s}$ → $\dot{x}$ → $\frac{1}{s}$ → x

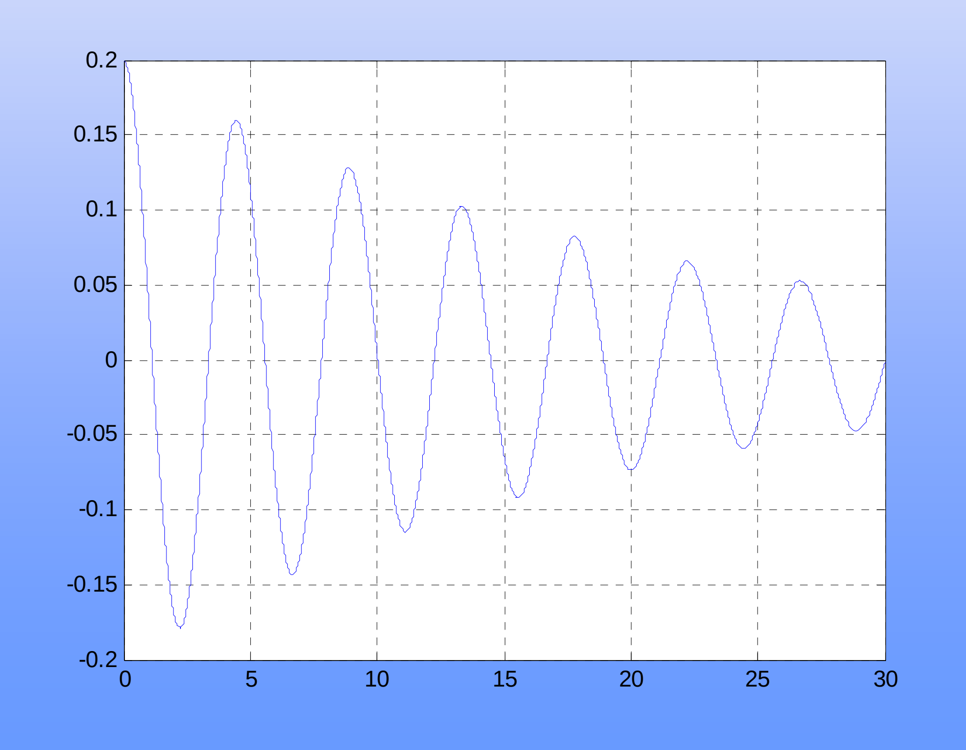
Scope

$$m\ddot{x}+c\dot{x}+kx=0,\quad x(0)=0.2,\quad \dot{x}(0)=0$$

$$m=0.5,\quad k=1,\quad c=0.05$$

动力方程变换为： $\ddot{x}=-\dfrac{c}{m}\dot{x}-\dfrac{k}{m}x \qquad \ddot{x}=-\dfrac{0.05}{0.5}\dot{x}-\dfrac{1}{0.5}x$



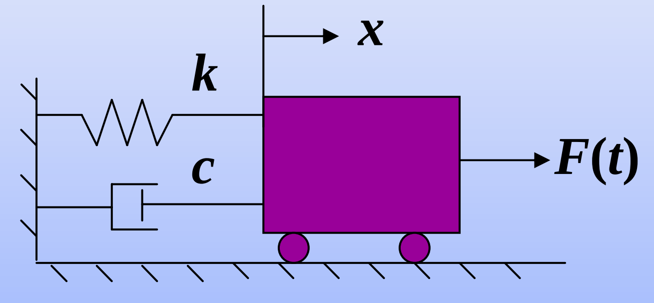


6. 传递函数模块

传递函数表示法频繁地应用于控制系统设计和系统的动态模拟。传递函数定义为系统在零初始条件下输出的 **Laplace** 变换与输入的 **Laplace** 变换之比。传递函数是一种描述系统动力学输入输出关系的简便方法。

例如：

考虑弹簧质量阻尼系统



动力学方程可写为：

$$m\ddot{x}+c\dot{x}+kx=F$$

作 **Laplace** 变换，并忽略初始条件，有：

$$ms^2X(s)+csX(s)+kX(s)=F(s)$$

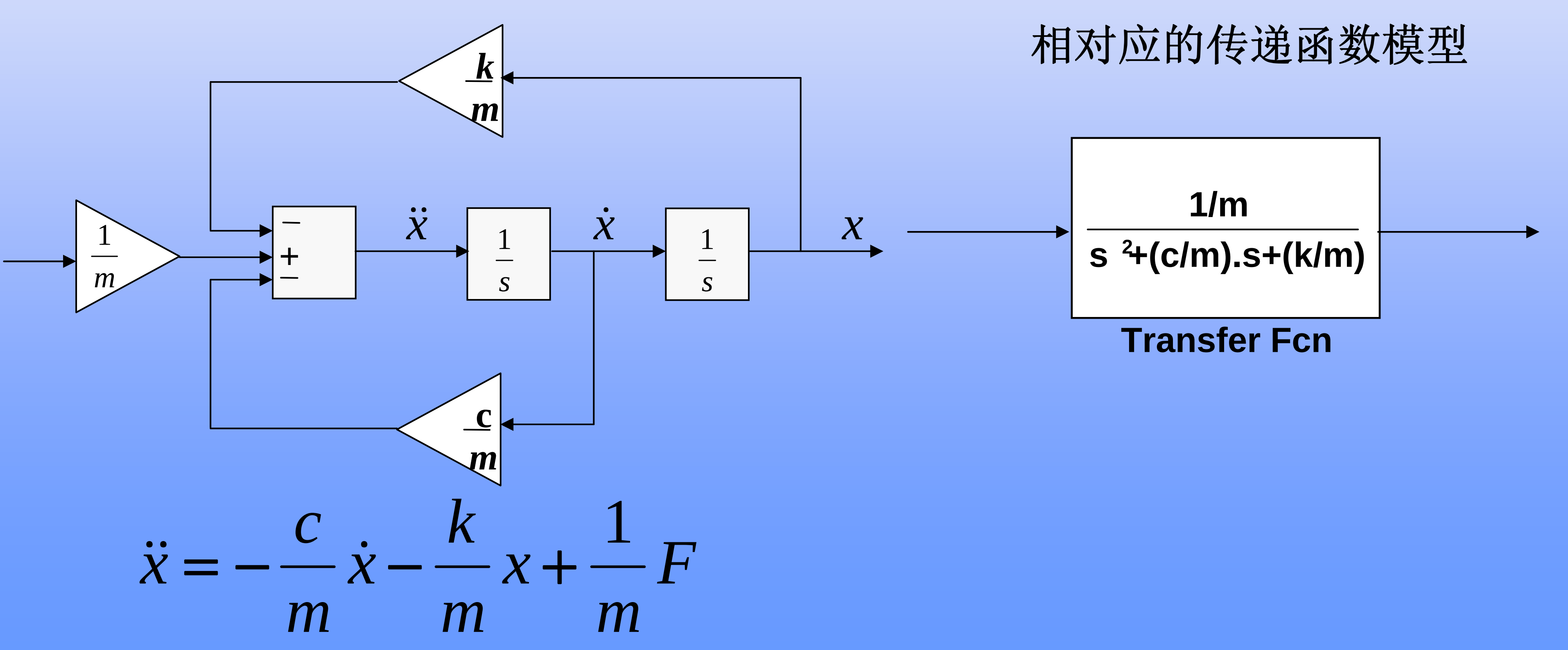
则系统的传递函数可表示为：

$$G(s)=\frac{X(s)}{F(s)}=\frac{1}{ms^2+cs+k}=\frac{1/m}{s^2+\frac{c}{m}s+\frac{k}{m}}$$

$$m\ddot{x}+c\dot{x}+kx=F$$

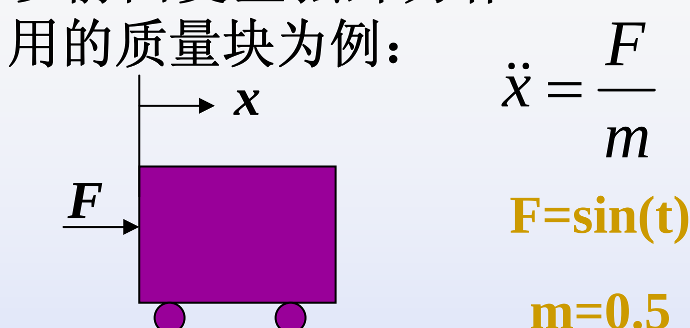
$$G(s)=\frac{X(s)}{F(s)}=\frac{1}{ms^2+cs+k}=\frac{1/m}{s^2+\frac{c}{m}s+\frac{k}{m}}$$

使用基本模块元素所建立起的系统模型



$$\ddot{x}=-\frac{c}{m}\dot{x}-\frac{k}{m}x+\frac{1}{m}F$$

以前面受正弦外力作用的质量块为例：

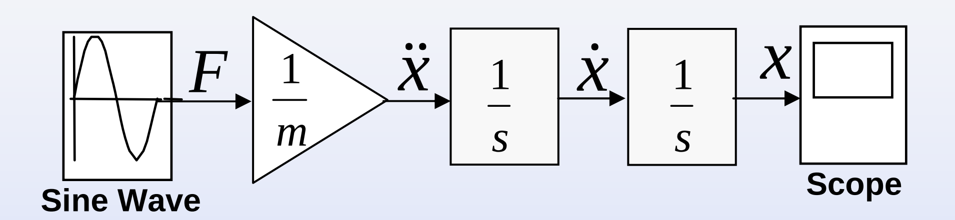


$$\ddot{x}=\frac{F}{m}$$

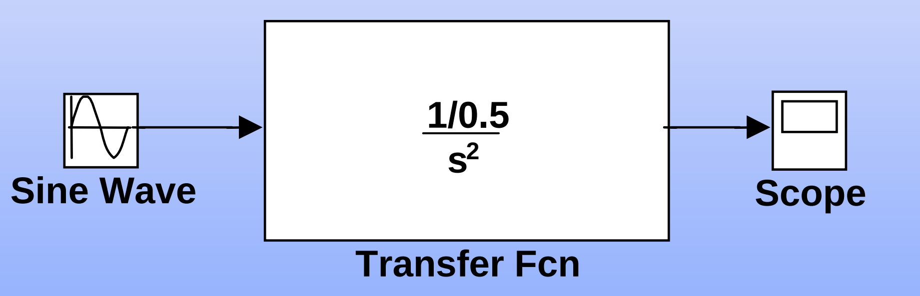
F=sin(t)

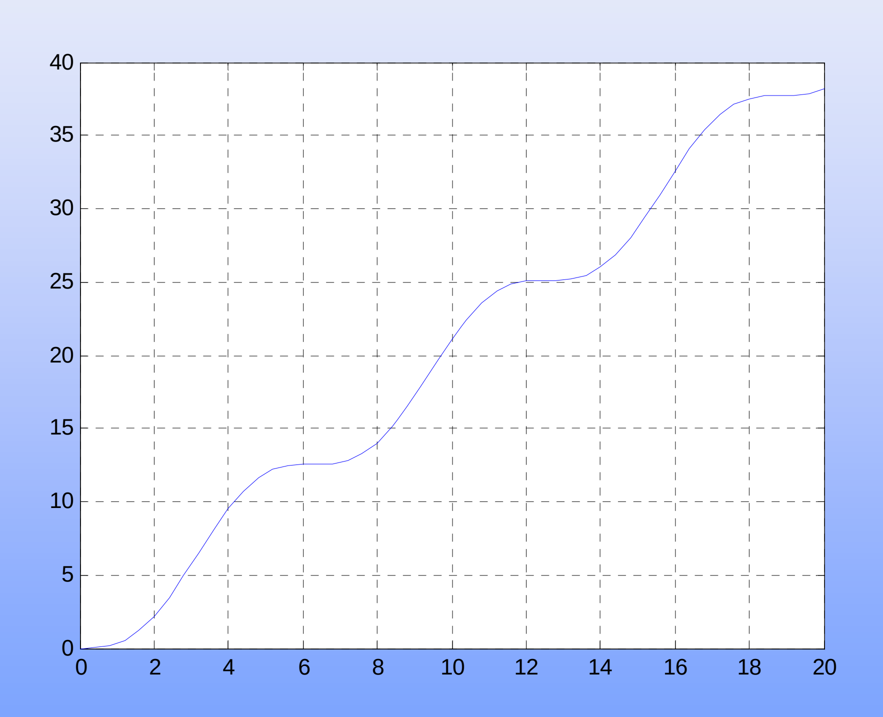
m=0.5

Simulink 模型仿真框图为：



传递函数仿真框图：





传递函数分子输入：**1/0.5**

分母输入：**[1, 0, 0]**

$$G(s)=\frac{X(s)}{F(s)}=\frac{1/m}{s^2+\frac{c}{m}s+\frac{k}{m}}$$

对传递函数模块的说明：

- 用传递函数模块对线性定常系统进行仿真，能够使仿真模型简单和紧凑，但是无法输出内部变量，如 **x** 的导数。

- 无法适用具有初始条件的情况。

- 传递函数模块只适用于单输入单输出系统，即单自由度系统，但无法应用于多输入多输出系统，即多自由度系统。

4.2 状态空间模块

状态空间模块可以起到与传递函数模块相同的作用。所不同的是，状态空间模块允许用户指定初始条件，并且可以共享内部变量。另外，状态空间模块可以用来模拟多输入多输出系统。

1. 状态空间的概念

考虑如下单自由度系统： $m\ddot{x}(t)+c\dot{x}(t)+kx(t)=F(t)$

x*(*t*)** 为系统的广义坐标；m***、***k***、***c*** 分别为单自由度系统的质量、刚度、阻尼系数；***F*** 为系统的外部输入。这些变量都是标量。

令： $\boldsymbol{Y}(t)=\begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix}$ 以 $x(t)$ 和 $\dot{x}(t)$ 构成的空间即为状态空间

Y*(*t*)** 为一个2维列向量，称为系统的状态向量。利用状态向量Y*(*t*)**，则上述动力方程可转化为：

$$\dot{\boldsymbol{Y}}(t)=\begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix}=\begin{bmatrix} 0 & 1 \\ -m^{-1}k & -m^{-1}c \end{bmatrix}\begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix}+\begin{bmatrix} 0 \\ m^{-1} \end{bmatrix}F(t)$$

$$\dot{\boldsymbol{Y}}(t)=\boldsymbol{A}\boldsymbol{Y}(t)+\boldsymbol{B}F(t)$$

输入矩阵

系统矩阵

考虑 **n** 自由度系统：

$$\boldsymbol{M}\ddot{\boldsymbol{X}}(t)+\boldsymbol{C}\dot{\boldsymbol{X}}(t)+\boldsymbol{K}\boldsymbol{X}(t)=\boldsymbol{F}(t)$$

$$\boldsymbol{X}(t)=[x_1(t),x_2(t),\cdots,x_n(t)]^T\in R^{n\times 1}$$ **n** 维广义坐标列向量

$\boldsymbol{M}\in R^{n\times n}$ 质量矩阵　　$\boldsymbol{C}\in R^{n\times n}$ 阻尼矩阵

$\boldsymbol{K}\in R^{n\times n}$ 刚度矩阵　　$\boldsymbol{F}(t)\in R^{n\times n}$ 外部输入列矩阵

令：

$$\boldsymbol{Y}(t)=\begin{bmatrix}\boldsymbol{X}(t)\\ \dot{\boldsymbol{X}}(t)\end{bmatrix}\in R^{2n\times 1}$$ **2n** 维系统状态列向量

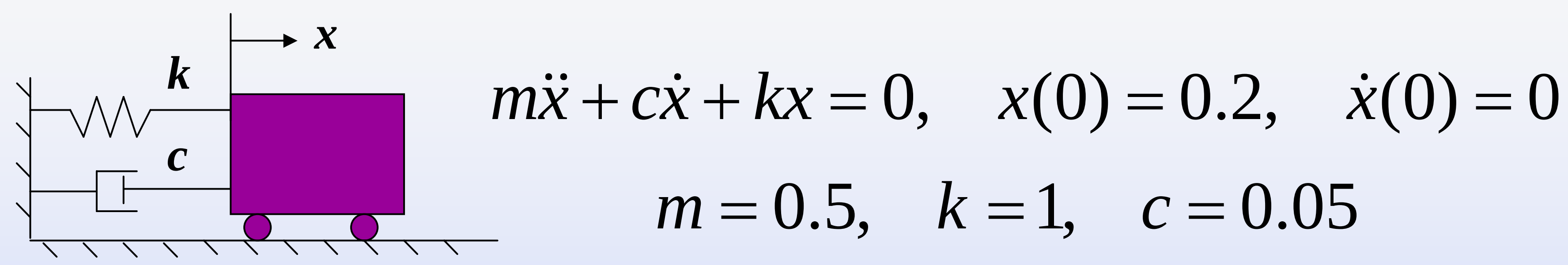
系统的状态方程：

$$\dot{\boldsymbol{Y}}(t)=\boldsymbol{A}\boldsymbol{Y}(t)+\boldsymbol{B}\boldsymbol{F}(t)$$

$$\boldsymbol{A}=\begin{bmatrix}\boldsymbol{0}_{n\times n} & \boldsymbol{I}_{n\times n}\\ -\boldsymbol{M}^{-1}\boldsymbol{K} & -\boldsymbol{M}^{-1}\boldsymbol{C}\end{bmatrix}\in R^{2n\times 2n}\qquad \boldsymbol{B}=\begin{bmatrix}\boldsymbol{0}_{n\times n}\\ \boldsymbol{M}^{-1}\end{bmatrix}\in R^{2n\times n}$$

系统矩阵　　输入矩阵

2. 线性单输入输出系统



$$m\ddot{x}+c\dot{x}+kx=0,\quad x(0)=0.2,\quad \dot{x}(0)=0$$

$$m=0.5,\quad k=1,\quad c=0.05$$

写成状态方程形式，有：　$\dot{\boldsymbol{Y}}(t)=\boldsymbol{A}\boldsymbol{Y}(t)$

$$\boldsymbol{A}=\begin{bmatrix}0 & 1\\ -m^{-1}k & -m^{-1}c\end{bmatrix}=\begin{bmatrix}0 & 1\\ -2 & -0.1\end{bmatrix}\qquad \boldsymbol{B}=\begin{bmatrix}0\\ m^{-1}\end{bmatrix}=\begin{bmatrix}0\\ 2\end{bmatrix}$$

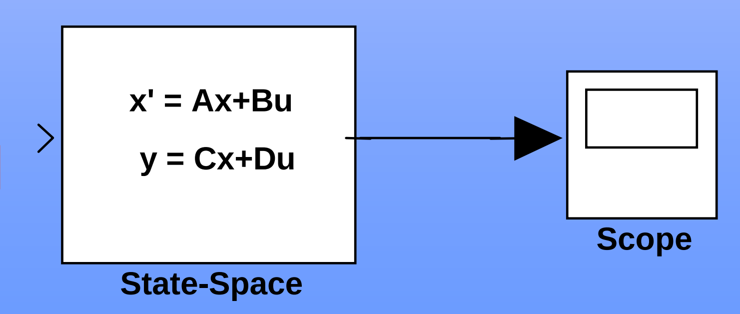
在状态空间模块中输入：

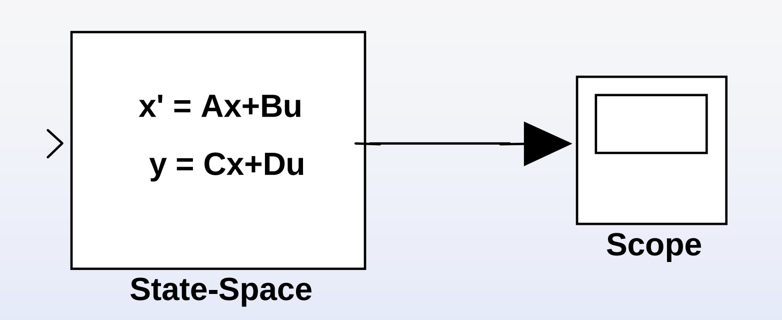
A=[0, 1; -2, -0.1]

B=[0; 2]　％因为没有外部输入，也可**B=[0; 0]**

C=[1, 1]　％ 不能写成**C=[0, 0]**，否则输出为**0**

D=0





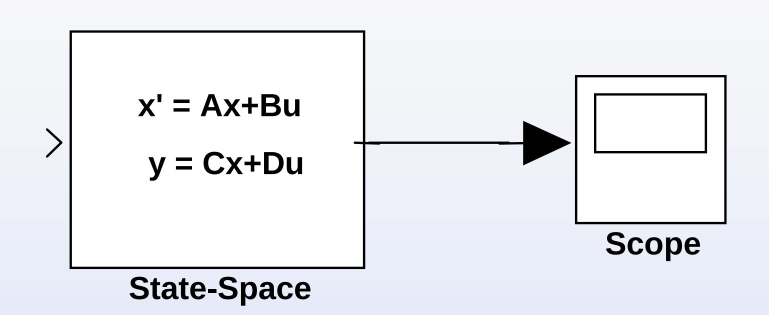
说明：对于本算例，因为考虑的是没有外部输入的自由振动，因此状态空间模块可以没有左端输入。

0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
0 5 10 15 20 25 30

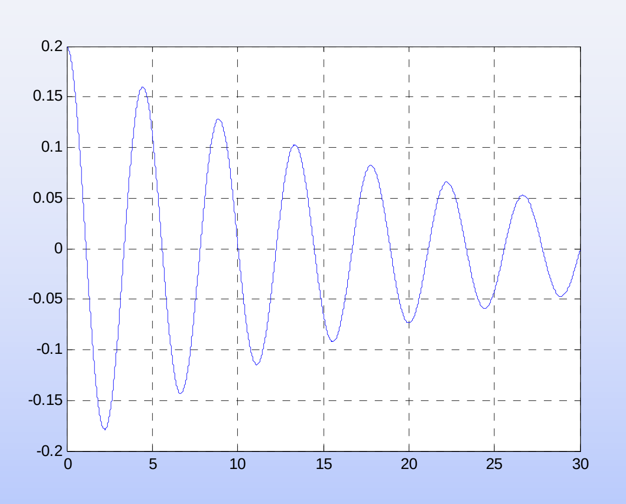
说明：左图是取**C=[1,0]**时的结果。若取**C=[1,1]**，则在**Scope**的图形中，将显示出位移和速度数据叠加后的数据的图形。

示波器输出为**y**的图形

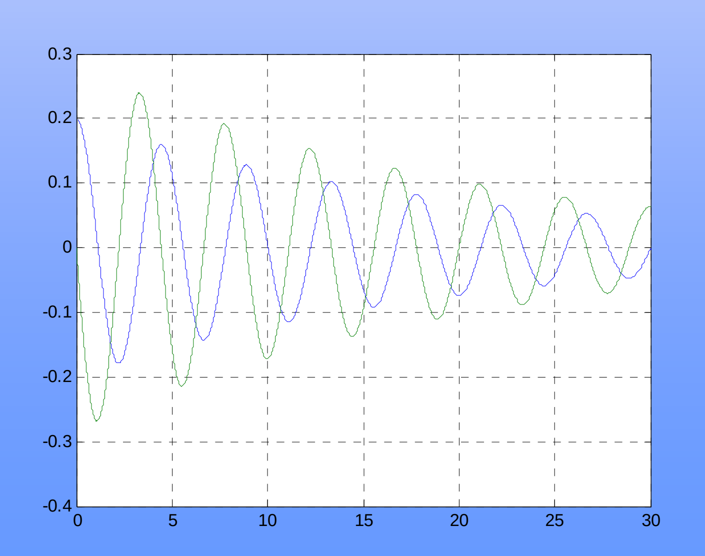
$$y = [1, 1]\begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$



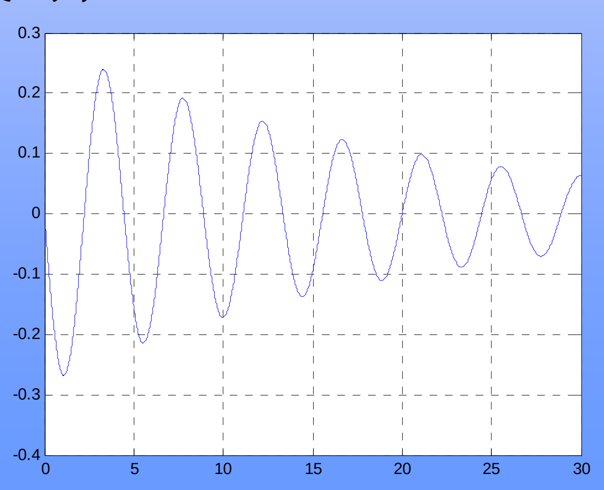
若取C=[1, 0]，则示波器显示出位移时程



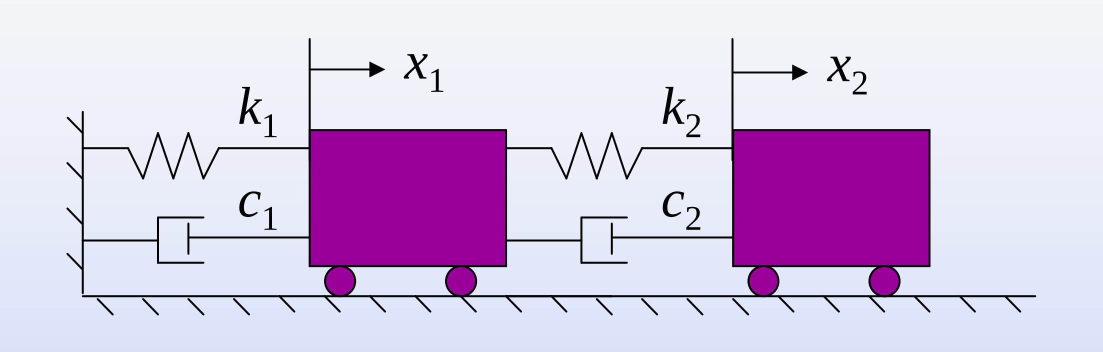
也可： C=[1,0;0,1]， D=zeros(2,1)
直接得出以下图形



若取C=[0, 1]，则示波器显示速度时程



3. 多输入输出系统



$$m_1 = 0.5, \quad m_2 = 1$$
$$k_1 = 1, \quad k_2 = 2$$
$$c_1 = c_2 = 0.05$$

$$\boldsymbol{M}\ddot{\boldsymbol{X}}(t) + \boldsymbol{C}\dot{\boldsymbol{X}}(t) + \boldsymbol{K}\boldsymbol{X}(t) = \boldsymbol{0} \qquad \boldsymbol{X}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

$$\boldsymbol{M} = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \qquad \boldsymbol{K} = \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \qquad \boldsymbol{C} = \begin{bmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_2 \end{bmatrix}$$

$$\boldsymbol{X}(0) = \begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0 \end{bmatrix} \qquad \dot{\boldsymbol{X}}(0) = \begin{bmatrix} \dot{x}_1(0) \\ \dot{x}_2(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$m_1 = 0.5, \quad m_2 = 1$$
$$k_1 = 1, \quad k_2 = 2$$
$$c_1 = c_2 = 0.05$$

$$\boldsymbol{M}\ddot{\boldsymbol{X}}(t) + \boldsymbol{C}\dot{\boldsymbol{X}}(t) + \boldsymbol{K}\boldsymbol{X}(t) = \boldsymbol{0} \qquad \boldsymbol{X}(0) = \begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0 \end{bmatrix} \qquad \dot{\boldsymbol{X}}(0) = \begin{bmatrix} \dot{x}_1(0) \\ \dot{x}_2(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

写成状态方程： $\dot{\boldsymbol{Y}}(t) = \boldsymbol{A}\boldsymbol{Y}(t)$ $\quad \boldsymbol{Y}(t) = \begin{bmatrix} \boldsymbol{X}(t) \\ \dot{\boldsymbol{X}}(t) \end{bmatrix} \quad \boldsymbol{A} = \begin{bmatrix} \boldsymbol{0}_{2\times2} & \boldsymbol{I}_{2\times2} \\ -\boldsymbol{M}^{-1}\boldsymbol{K} & -\boldsymbol{M}^{-1}\boldsymbol{C} \end{bmatrix}$

初始条件：

$$\boldsymbol{Y}(0) = \begin{bmatrix} \boldsymbol{X}(0) \\ \dot{\boldsymbol{X}}(0) \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

系统矩阵：

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -6 & 4 & -0.2 & 0.1 \\ 2 & -2 & 0.05 & -0.05 \end{bmatrix}$$

$$\boldsymbol{M}\ddot{\boldsymbol{X}}(t)+\boldsymbol{C}\dot{\boldsymbol{X}}(t)+\boldsymbol{K}\boldsymbol{X}(t)=\boldsymbol{0}$$

$$\dot{\boldsymbol{Y}}(t)=\boldsymbol{A}\boldsymbol{Y}(t)$$

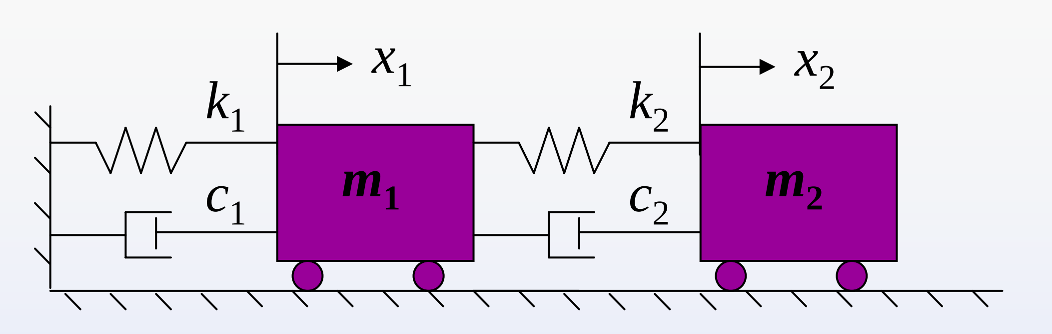
$$\boldsymbol{A}=\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -6 & 4 & -0.2 & 0.1 \\ 2 & -2 & 0.05 & -0.05 \end{bmatrix} \qquad \boldsymbol{Y}(0)=\begin{bmatrix} 0.2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

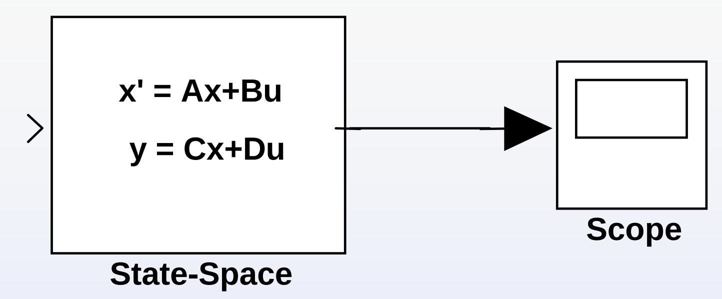
在状态空间模块中输入：**A=[0,0,1,0;0,0,0,1;-6,4,-0.2,0.1;2,-2,0.05,-0.05]**

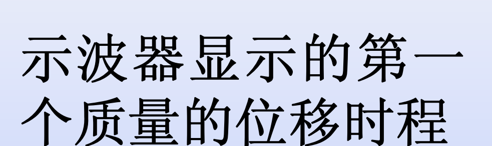
B=[0;0;0;0] %因为没有外部输入

C=[1,1,1,1] % 不能写成**C=[0,0,0, 0]**,否则输出为**0**

D=0

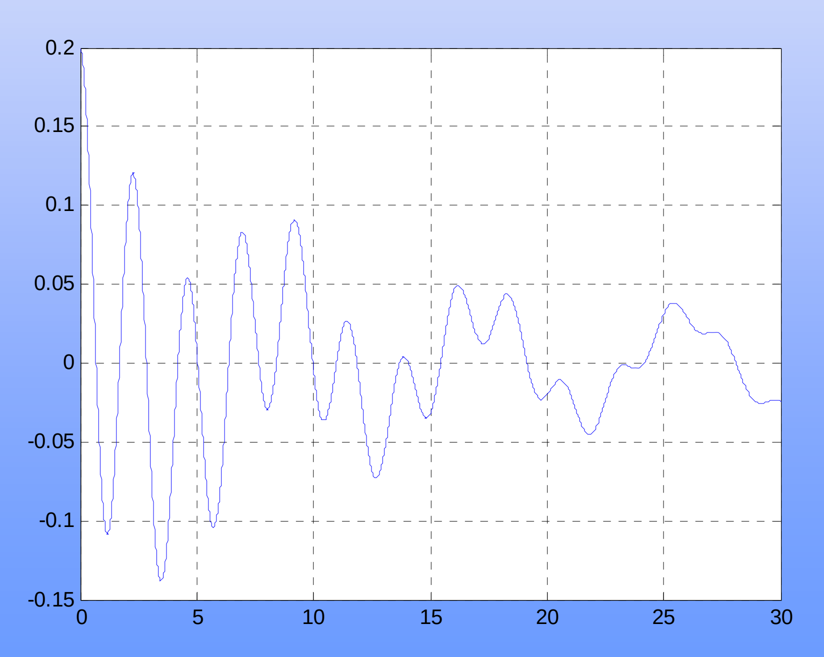






示波器显示的第一个质量的位移时程

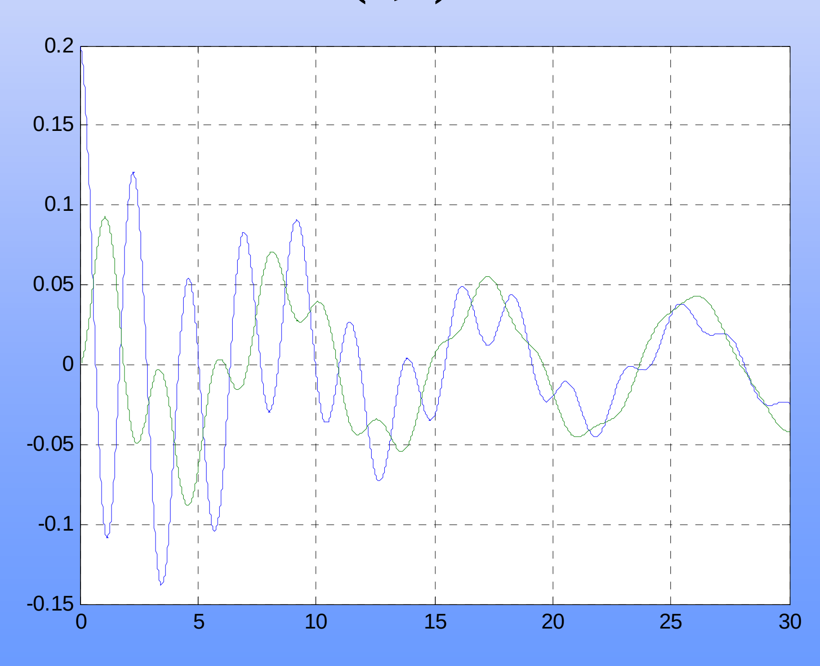
C=[1,0,0,0]

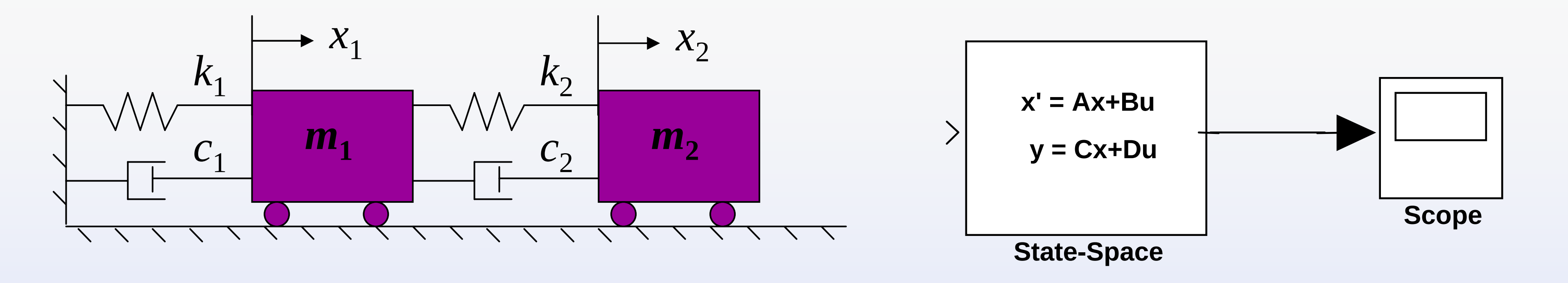


在**MATLAB**工作空间画出的两个质量的位移时程

也可：**C=[1,0,0,0;0,1,0,0]**

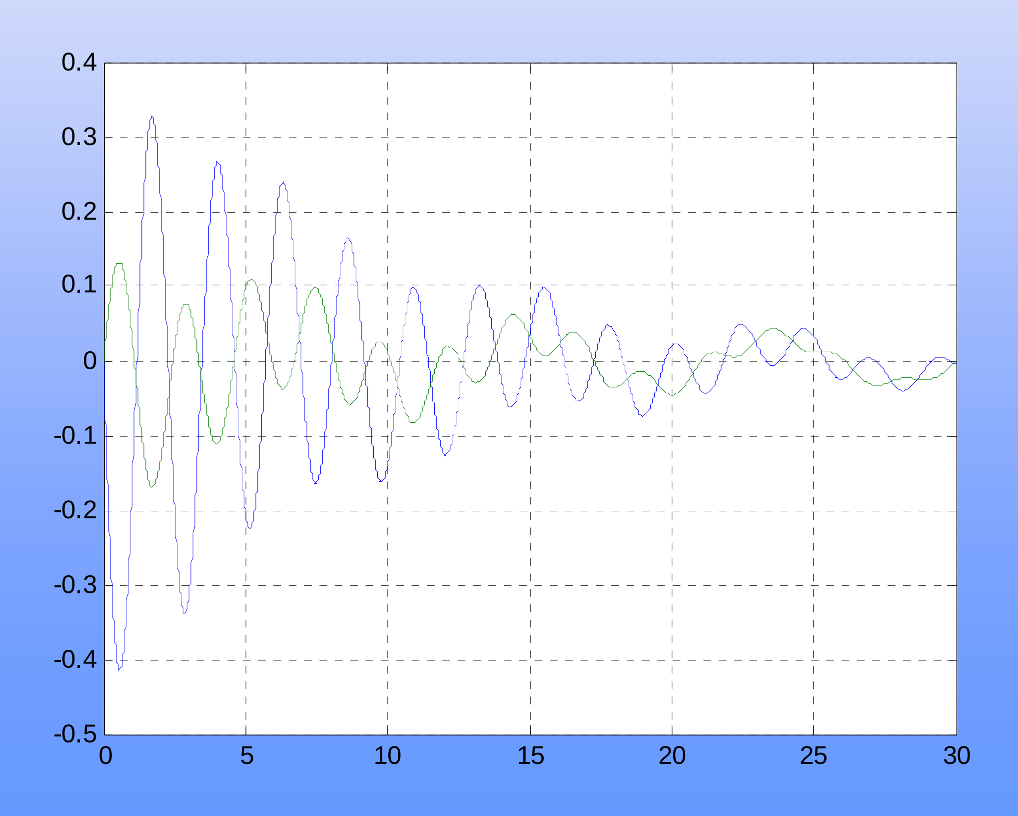
D=zeros(4,1)直接得出





在MATLAB工作空间画出的两个质量的速度时程

也可：C=[0,0,1,0;0,0,0,1]，D=zeros(4,1)直接得出



4.3 离散系统模块

离散系统采用差分方程表示。离散信号是一组以采样周期为间隔的离散时间序列。对于大多数的物理系统，信号原本都是在时间上连续的。但在对信号的采集过程中，需要通过传感器等采集工具对信号进行采集，因此最终得到的信号都是以采样周期为时间间隔的离散数据。若原物理系统也用离散数据形式进行描述，则构成了离散时间系统。

1. 离散增益模块
2. 离散求和模块

} 与连续系统采用的模块相同

3. 离散延迟模块
4. 时间离散积分模块
5. 简单离散系统模型
6. 离散传递函数模块

4.4 离散状态空间模块

1. 连续时间系统的离散化

考虑 n 自由度线性定常系统： $\dot{\boldsymbol{Y}}(t)=\boldsymbol{A}\boldsymbol{Y}(t)+\boldsymbol{B}\boldsymbol{u}(t)$ （1）

其中： $\boldsymbol{Y}(t)\in R^{n\times 1}$， $\boldsymbol{A}\in R^{n\times n}$， $\boldsymbol{B}\in R^{n\times r}$， $\boldsymbol{u}(t)\in R^{r\times 1}$

式（1）的解析解： $\boldsymbol{Y}(t)=e^{\boldsymbol{A}(t-t_0)}\boldsymbol{Y}(t_0)+\int_{t_0}^{t}e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau$ （2）

令： $t_0=kT, t=(k+1)T$ 其中：T 为数据采样周期

并采用如下零阶保持器： $\boldsymbol{u}(t)=\boldsymbol{u}(k),\quad kT\le t<(k+1)T$ （3）

则式（3）转化为： $\boldsymbol{Y}(k+1)=e^{\boldsymbol{A}T}\boldsymbol{Y}(k)+\int_{kT}^{(k+1)T}e^{\boldsymbol{A}[(k+1)T-\tau]}d\tau\boldsymbol{B}\boldsymbol{u}(k)$

令： $t=(k+1)T-\tau$

则上式变为： $\boldsymbol{Y}(k+1)=e^{\boldsymbol{A}T}\boldsymbol{Y}(k)+\int_{0}^{T}e^{\boldsymbol{A}t}dt\boldsymbol{B}\boldsymbol{u}(k)$ （4）

$$\dot{\boldsymbol{Y}}(t)=\boldsymbol{AY}(t)+\boldsymbol{Bu}(t) \qquad (1)$$

$$\boldsymbol{Y}(k+1)=e^{\boldsymbol{A}T}\boldsymbol{Y}(k)+\int_0^T e^{\boldsymbol{A}t}dt\boldsymbol{Bu}(k) \qquad (4)$$

令：$$\boldsymbol{F}=e^{\boldsymbol{A}T}\in R^{n\times n},\quad \boldsymbol{G}=\int_0^T e^{\boldsymbol{A}t}dt\boldsymbol{B}\in R^{n\times r} \qquad (5)$$

则式（4）转化为：$$\boldsymbol{Y}(k+1)=\boldsymbol{FY}(k)+\boldsymbol{Gu}(k) \qquad (6)$$

式（5）中的两个变量将于有限时间内趋于常值矩阵。

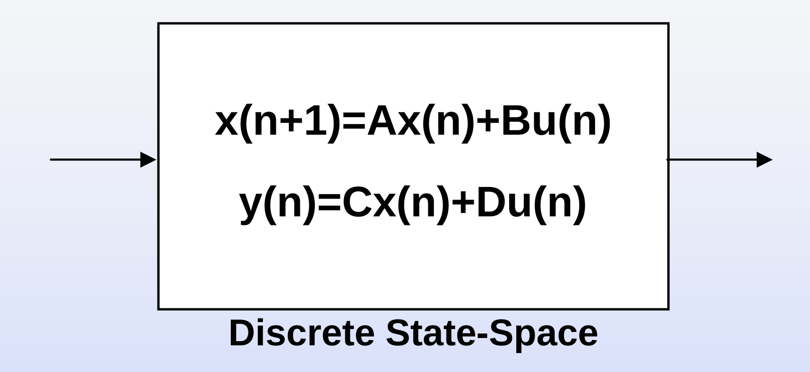
这样，式（1）所示的连续时间系统就转换成了式（6）所示的离散时间系统。矩阵 $\boldsymbol{F}$ 称为离散时间系统的系统矩阵，$\boldsymbol{G}$ 称为离散时间系统的输入矩阵。

2. $\boldsymbol{F}$ 和 $\boldsymbol{G}$ 的计算

根据矩阵指数的定义，有：$$\boldsymbol{F}=e^{\boldsymbol{A}T}=I+\boldsymbol{A}T+\frac{\boldsymbol{A}^2T^2}{2!}+\cdots=\sum_{k=0}^{\infty}\frac{\boldsymbol{A}^kT^k}{k!}$$

同时不难求得：$$\boldsymbol{G}=\int_0^T e^{\boldsymbol{A}t}dt\boldsymbol{B}=\sum_{k=0}^{\infty}\frac{\boldsymbol{A}^kT^{k+1}}{(k+1)!}\boldsymbol{B}=\sum_{k=1}^{\infty}\frac{\boldsymbol{A}^{k-1}T^k}{k!}\boldsymbol{B}$$

3. 离散状态空间模块



4. 采用离散状态空间模块进行仿真

对于两自由度系统：

$$\boldsymbol{M\ddot{X}}(t)+\boldsymbol{C\dot{X}}(t)+\boldsymbol{KX}(t)=\boldsymbol{0}$$

$$\boldsymbol{X}(0)=[0.2 \quad 0]^T \quad \boldsymbol{\dot{X}}(0)=[0 \quad 0]^T$$

k_1 x_1 m_1 c_1 k_2 x_2 m_2 c_2

$$\boldsymbol{X}(t)=\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \quad \boldsymbol{M}=\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \quad \boldsymbol{K}=\begin{bmatrix} k_1+k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \quad \boldsymbol{C}=\begin{bmatrix} c_1+c_2 & -c_2 \\ -c_2 & c_2 \end{bmatrix}$$

$$M\ddot{X}(t)+C\dot{X}(t)+KX(t)=0 \qquad X(0)=[0.2 \quad 0]^T \quad \dot{X}(0)=[0 \quad 0]^T$$

写成状态方程： $\dot{Y}(t)=AY(t)$ $\quad Y(t)=\begin{bmatrix} X(t) \\ \dot{X}(t) \end{bmatrix} \quad A=\begin{bmatrix} 0_{2\times2} & I_{2\times2} \\ -M^{-1}K & -M^{-1}C \end{bmatrix}$

初始条件： $Y(0)=\begin{bmatrix} X(0) \\ \dot{X}(0) \end{bmatrix}=[0.2 \quad 0 \quad 0 \quad 0]^T$

连续时间系统的系统矩阵：

$$A=\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -6 & 4 & -0.2 & 0.1 \\ 2 & -2 & 0.05 & -0.05 \end{bmatrix}$$

离散时间系统的系统矩阵（假定采样周期为 **T＝0.02**）

(程序名：chap5_section53.m)

$$F=\begin{bmatrix} 0.9988 & 0.0008 & 0.02 & 0 \\ 0.0004 & 0.9996 & 0 & 0.02 \\ -0.1197 & 0.0798 & 0.9948 & 0.0028 \\ 0.0399 & -0.0399 & 0.0014 & 0.9986 \end{bmatrix}$$

$$Y(k+1)=FY(k) \qquad Y(0)=[0.2 \quad 0 \quad 0 \quad 0]^T$$

$$Y(k+1)=FY(k)$$

$$Y(0)=[0.2 \quad 0 \quad 0 \quad 0]^T$$

$$F=\begin{bmatrix} 0.9988 & 0.0008 & 0.02 & 0 \\ 0.0004 & 0.9996 & 0 & 0.02 \\ -0.1197 & 0.0798 & 0.9948 & 0.0028 \\ 0.0399 & -0.0399 & 0.0014 & 0.9986 \end{bmatrix}$$

在状态空间模块中输入： **A=[0.9988,0.0008,0.02,0;0.0004,0.9996,0,0.02;-0.1197,0.0798,0.9948,0.0028;0.0399,-0.0399,0.0014,0.99865]**

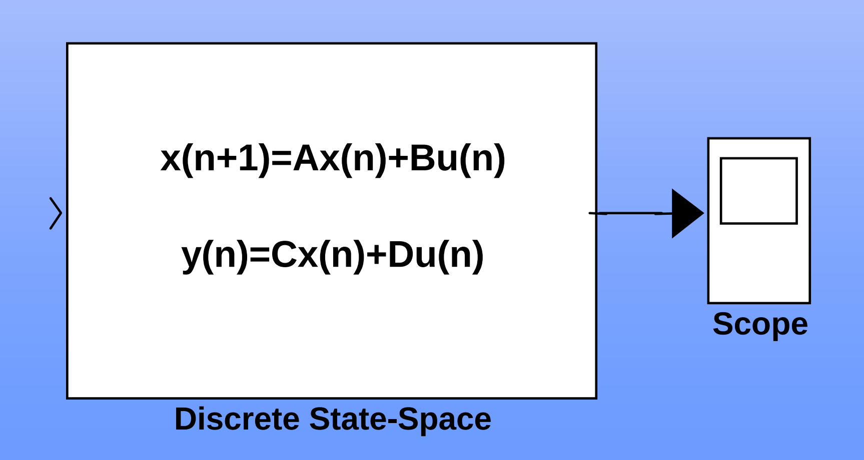
B=[0;0;0;0] %因为没有外部输入

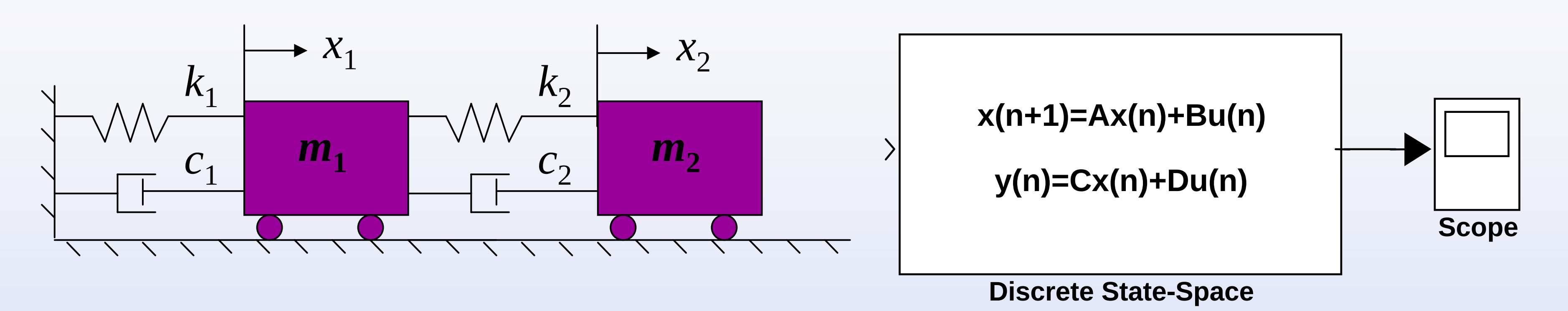
C=[1,1,1,1] % 不能写成**C=[0,0,0,0],**否则输出为**0**

D=0

初值输入为：**[0.2; 0; 0; 0]**

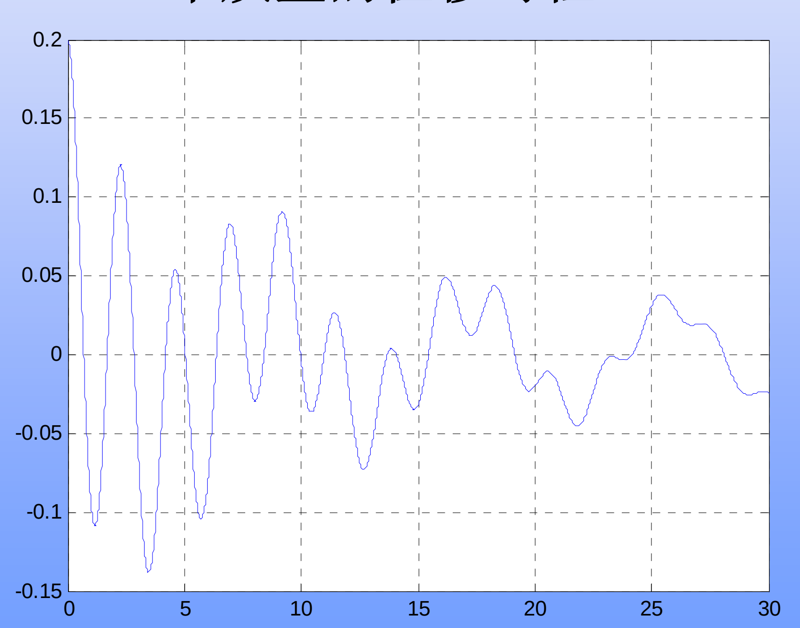
步长输入为：**0.02**

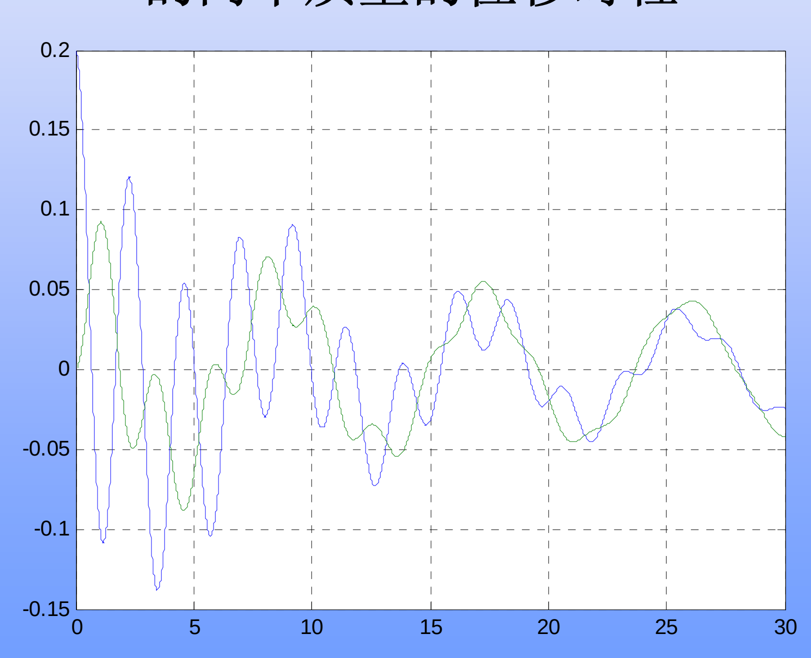




示波器显示的第一个质量的位移时程

在MATLAB工作空间画出的两个质量的位移时程





与前面连续时间系统的仿真结果一致。

SIMULINK (5)

连续系统

本章学习内容和目的

- 进一步掌握连续系统的建模和仿真技巧
- 掌握向量线性系统的描述方法
- 结合蹦极跳的算例领会复杂系统的建模方法与仿真

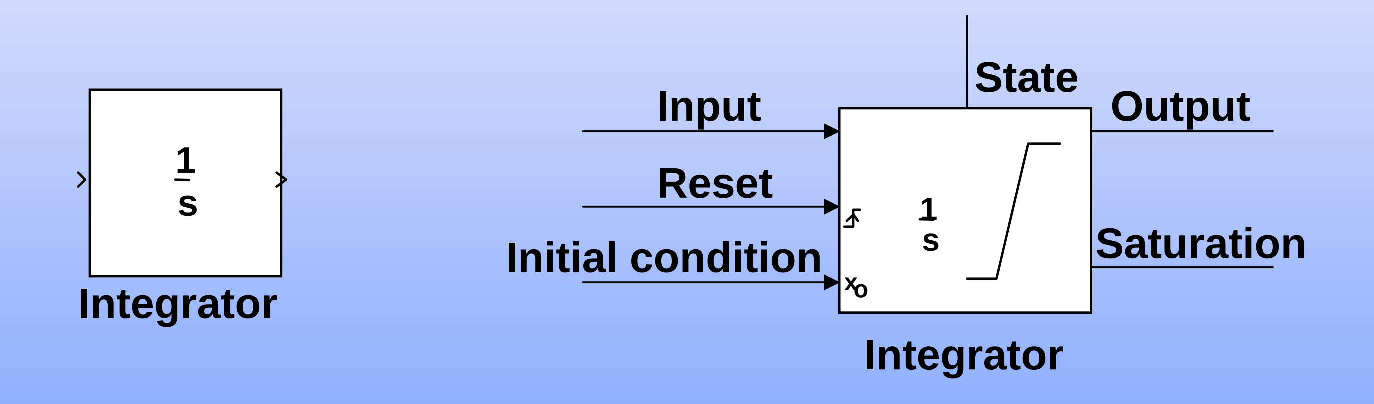
积分模块

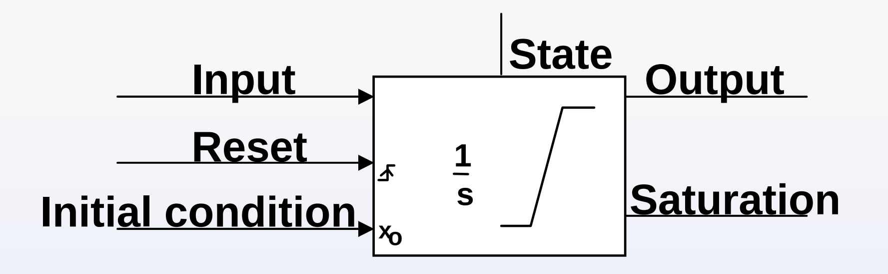
高级积分器

在使用 **Simulink** 对实际的动态系统进行仿真时，积分运算可以说是 **Simulink** 求解器的核心技术之一。在前面章节中使用过积分模块对数据进行积分处理，事实上我们所使用的是积分模块的一种简单积分方式。积分模块还有一种积分方式——重置积分方式。所谓的重置积分方式是指当重置信号触发时将模块的输出重置为初始条件。本节积分模块的重置积分方式——高级积分器进行介绍。

首先对积分器的各个端口进行简单的介绍。

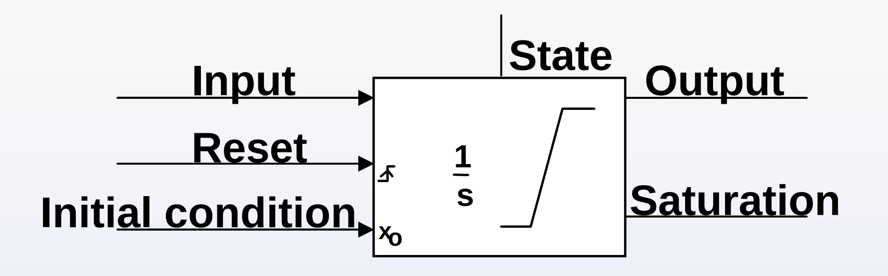
下图分别为使用缺省参数设置下的积分器外观与选择所用参数设置下的积分器的外观比较。





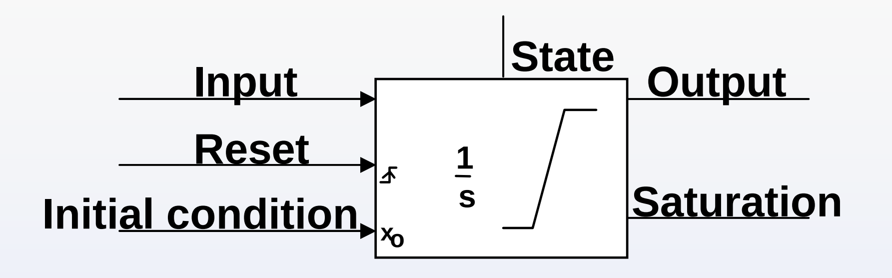
【External reset】为外部重置设置。它用在当重置信号发生触发事件时，模块将按照初始条件重置状态量。可以采用不同的触发方式对积分器状态进行重置：

- **none**：关闭外部重置；
- **rising**：当模块接收到的触发信号上升通过零点时，重置过程开始；
- **falling**：当模块接收到的触发信号下降通过零点时，重置过程开始；
- **either**：无论触发信号上升或下降通过零点，重置过程都开始；
- **level**：当触发信号非零时，使得积分器输出保持在初始状态。



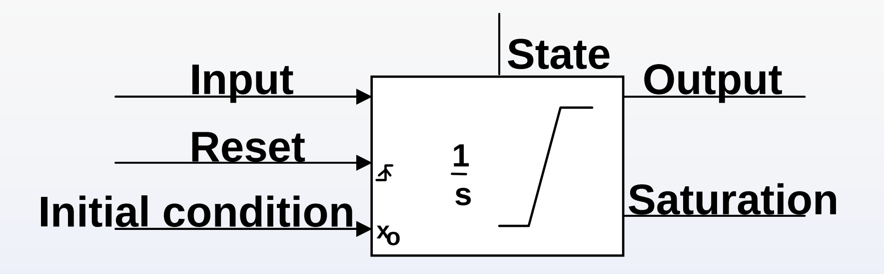
【Initial condition source】为初始条件设置。设置积分器初始条件的方法有两种：

- **external：** 从外部输入源设置初始条件。初始条件设置端口以 **x0** 作为标志；
- **internal：** 在积分器模块参数对话框中设置初始条件，说明模块的初始值是从内部获得的。 选上后，下面将出现要求输入初始值的输入栏。**Internal** 为默认设置。



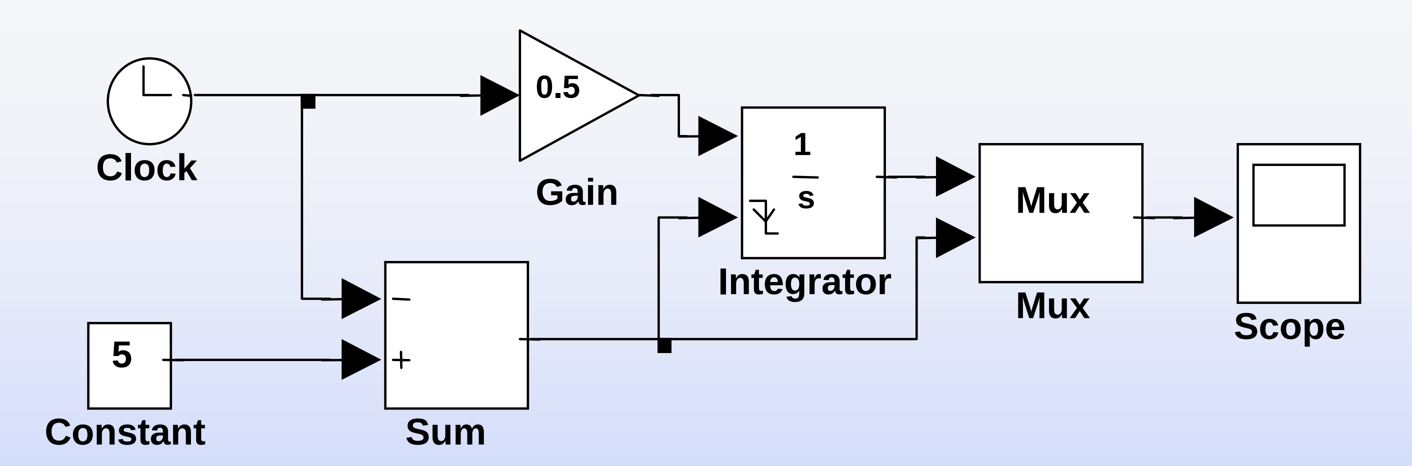
【**Limit output**】积分器输出范围限制。在某些情况下，积分器的输出可能会超出系统本身所允许的上限或下限值，选择积分器输出范围限制框（**Limit output**）并设置上限值（**Upper saturation limit**）与下限值（**Lower saturation limit**），可以将积分器的输出限制在一个给定的范围之内。此时积分器的输出服从下面的规则：

- 当积分结果小于或等于下限值并且输入信号为负，积分器的输出保持在下限值（下饱和区）；
- 当积分结果在上限值与下限值之间时，积分器输出为实际积分结果；
- 当积分结果大于或等于上限值并且输入信号为正，积分器的输出保持在上限值（上饱和区）。



【Show saturation port】在积分器中显示饱和端口。此端口位于输出端口的下方。饱和端口的输出有三种情况，用来表示积分器的饱和状态：

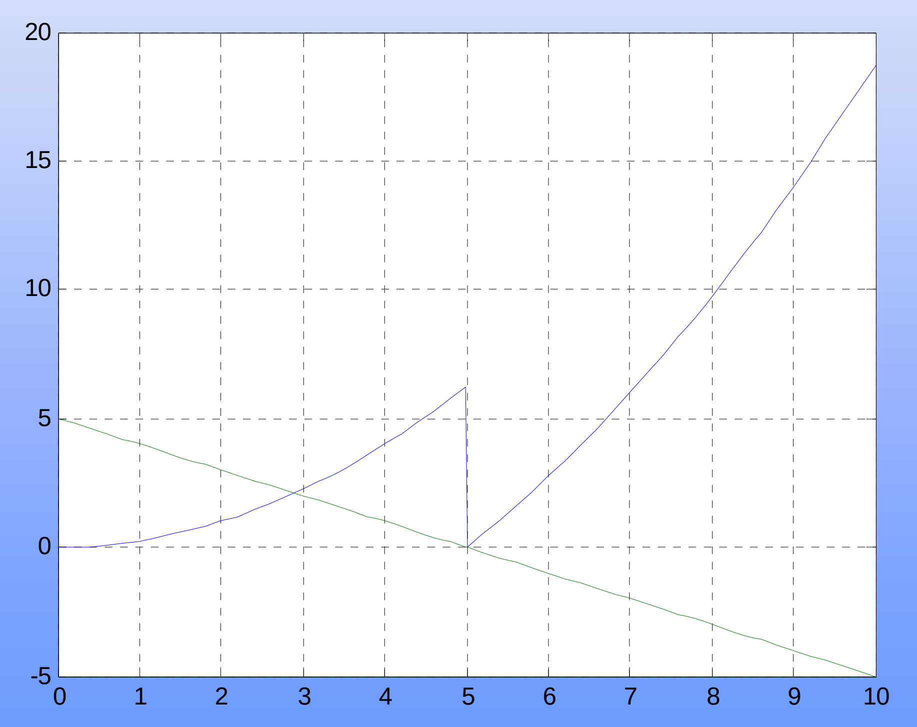
- 输出为 **1**，表示积分器处于上饱和区；
- 输出为 **0**，表示积分器处于正常范围之内；
- 输出为 **－1**，表示积分器处于下饱和区。



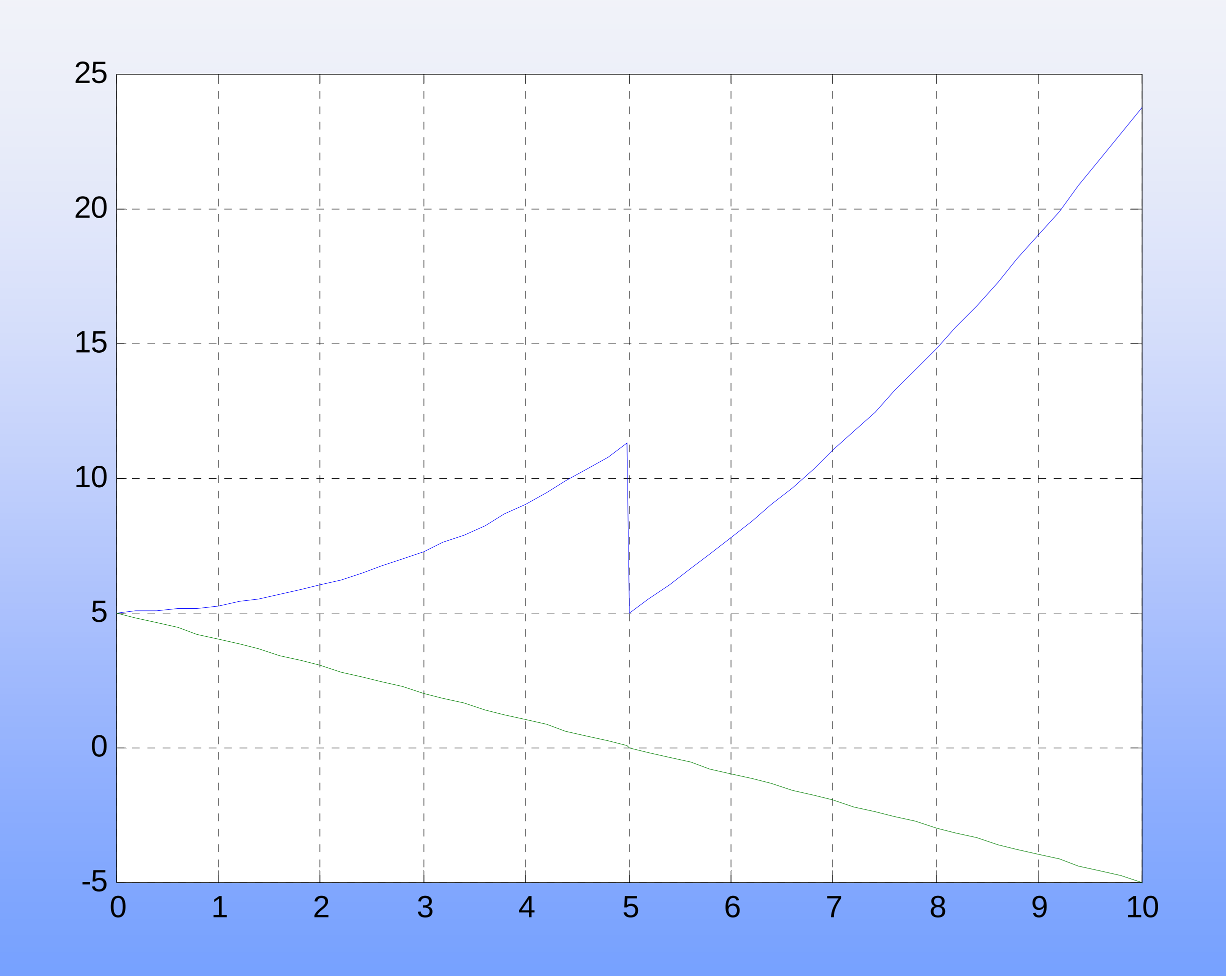
说明：

（1）积分器的初始值为0；

（2）【External reset】的设置为：falling。

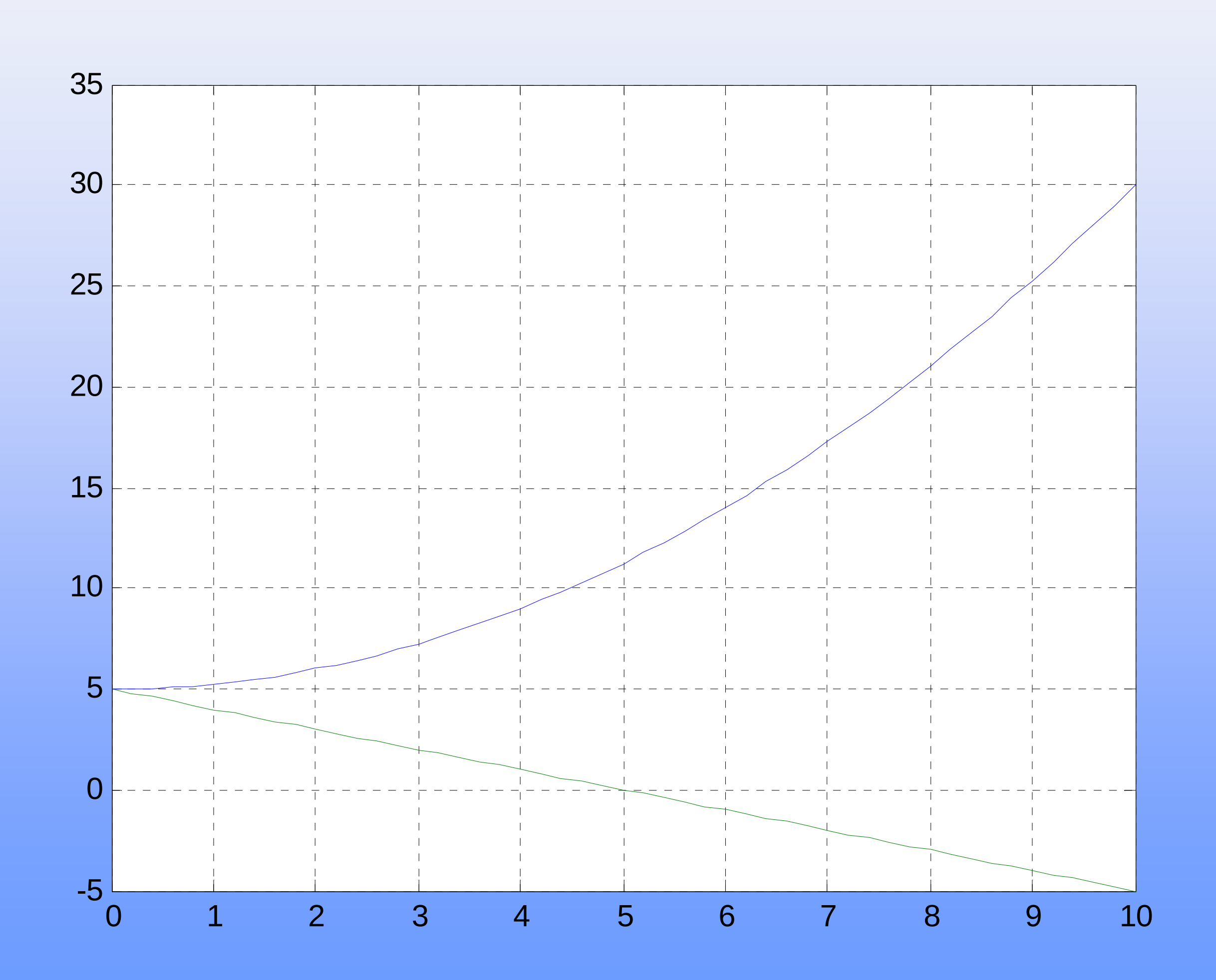


如果将初始值设置为**5**，则结果为：



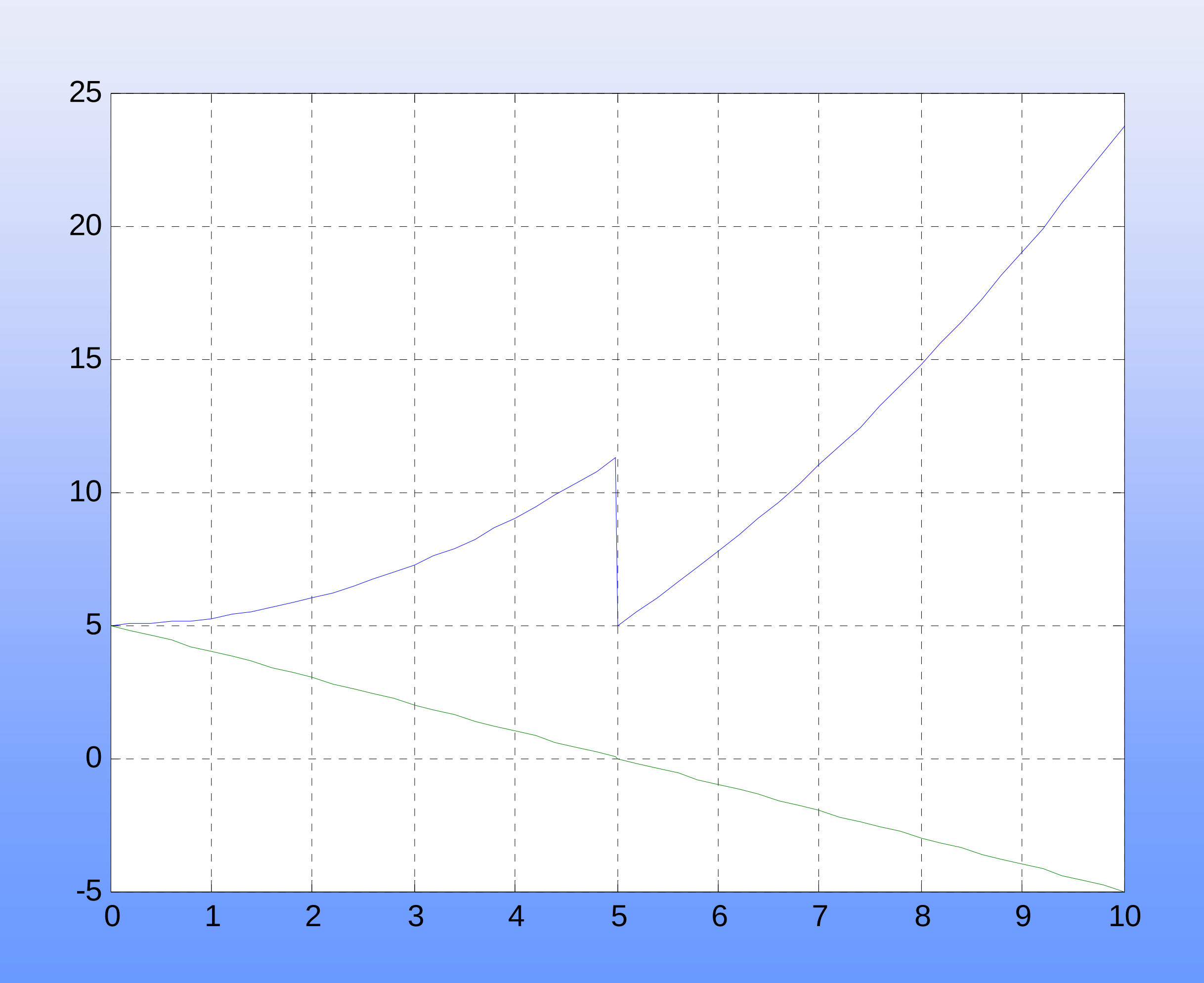
（**1**）积分器的初始值为**5**；

（**2**）【**External reset**】的设置为：**rising**。



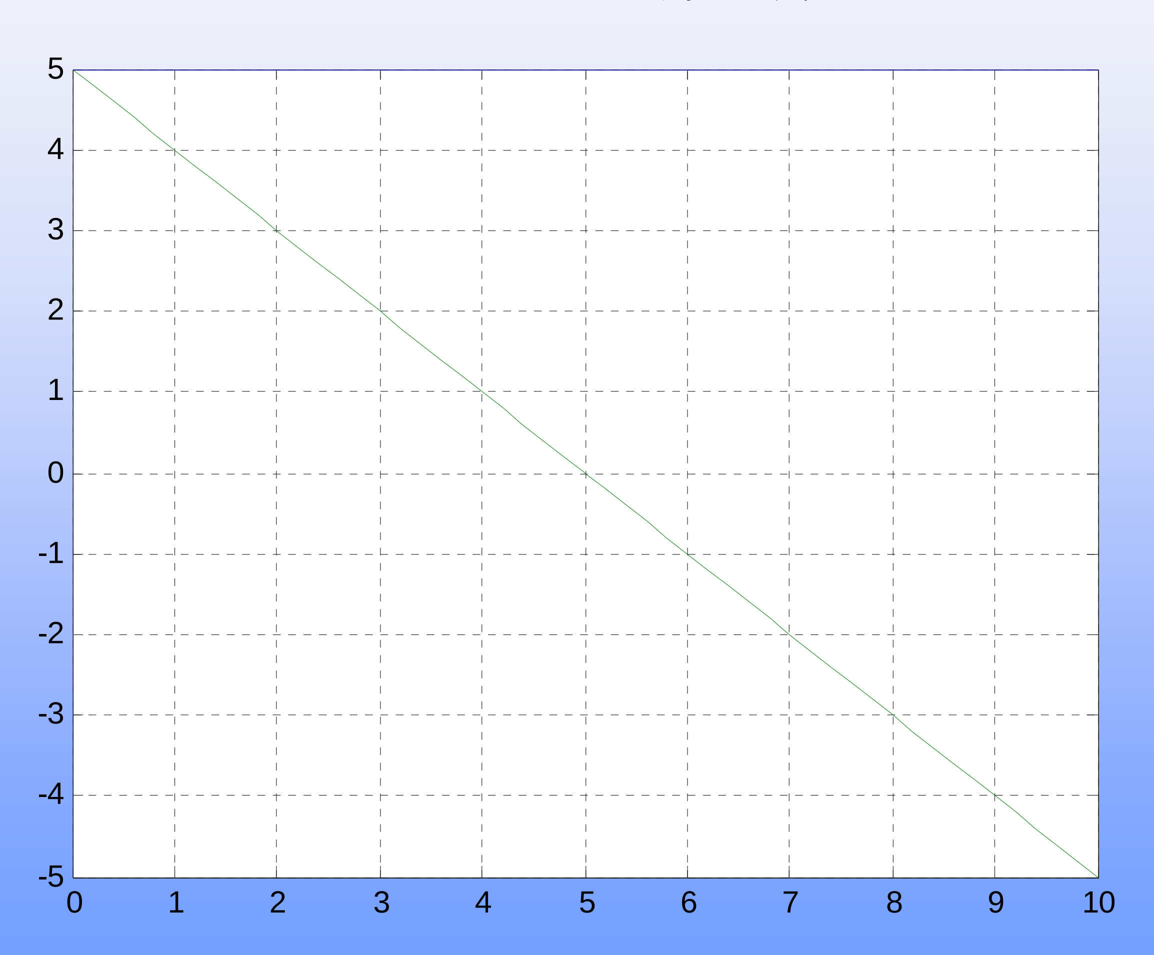
（**1**）积分器的初始值为**5**；

（**2**）【**External reset**】的设置为：**either**。

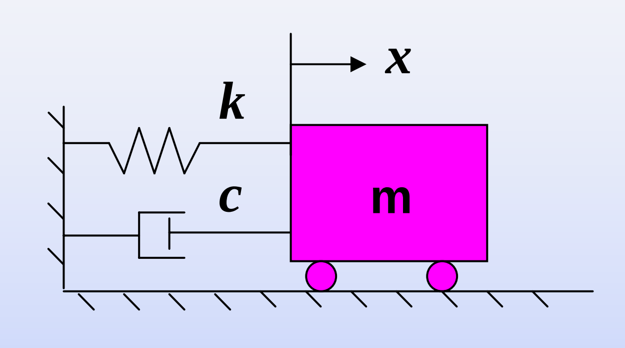


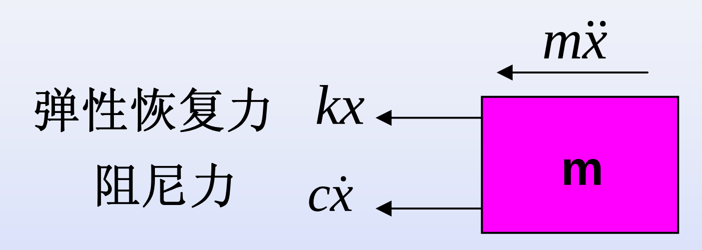
（1）积分器的初始值为**5**；

（2）【**External reset**】的设置为：**level**。



连续系统仿真（采用积分模块）





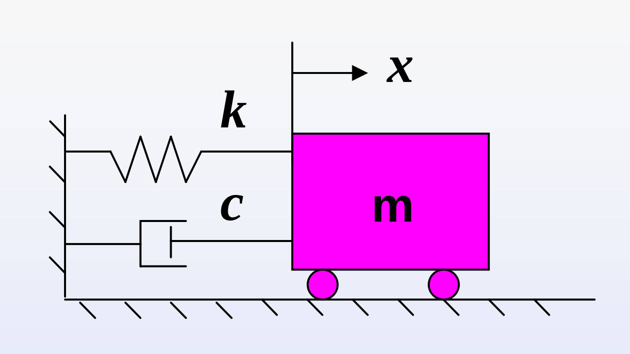
系统动力方程： $m\ddot{x}+c\dot{x}+kx=0$

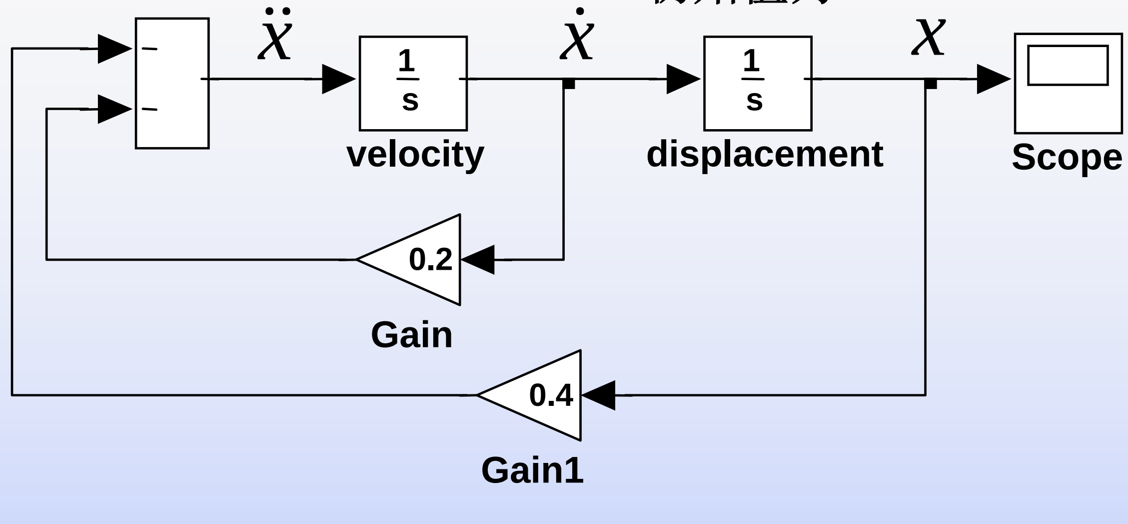
系统参数： $m=5,\quad k=2,\quad c=1$

对原动力方程作变换： $\ddot{x}=-\dfrac{c}{m}\dot{x}-\dfrac{k}{m}x$

代入参数： $\ddot{x}=-0.2\dot{x}-0.4x$

假定有如下初始条件： $x(0)=1,\quad \dot{x}(0)=0$



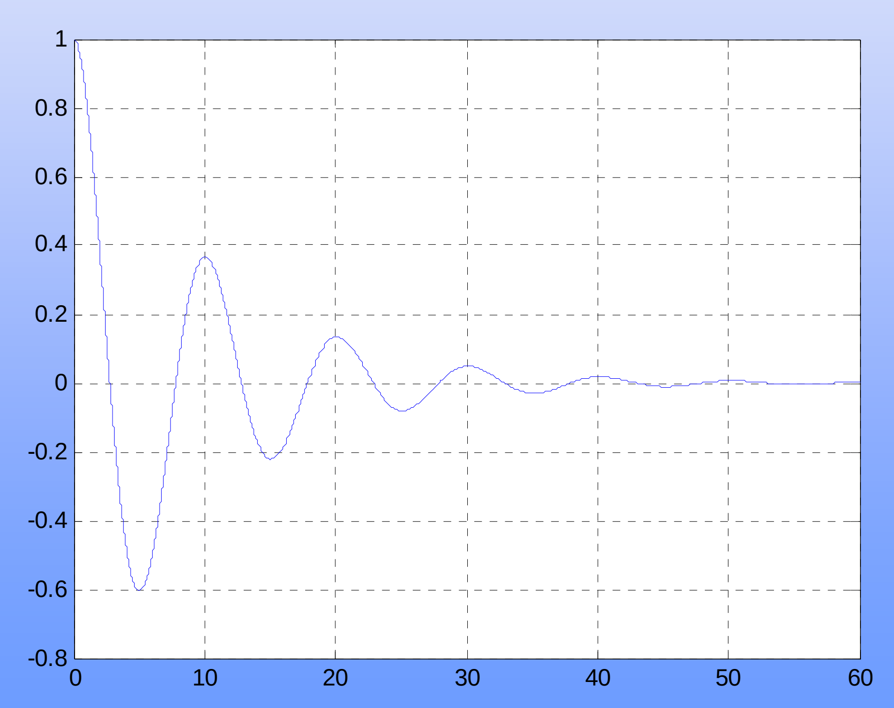


$$\ddot{x} = -\frac{c}{m}\dot{x} - \frac{k}{m}x$$

$$m = 5, \quad k = 2, \quad c = 1$$

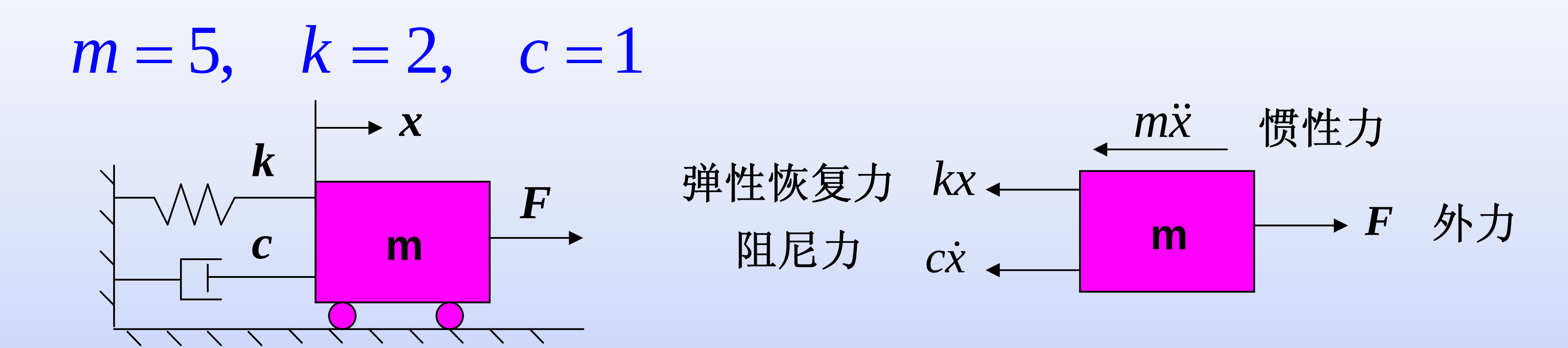
$$\ddot{x} = -0.2\dot{x} - 0.4x$$

$$x(0) = 1, \quad \dot{x}(0) = 0$$



连续系统仿真（采用传递函数模块）

$m = 5, \quad k = 2, \quad c = 1$



系统动力方程： $m\ddot{x} + c\dot{x} + kx = F$

经过 **Laplace** 变换，有： $ms^2X(s) + csX(s) + kX(s) = F(s)$

传递函数为：

$$G(s) = \frac{X(s)}{F(s)} = \frac{1}{ms^2 + cs + k} = \frac{1/m}{s^2 + \frac{c}{m}s + \frac{k}{m}} = \frac{0.2}{s^2 + 0.2s + 0.4}$$

已知条件：假设系统处于零平衡位置，外力函数为幅值为 **1** 的阶跃函数。

1

Step Gain2 0.2 velocity $\frac{1}{s}$ displacement $\frac{1}{s}$ Scope Gain 0.2 Gain1 0.4

$\ddot{x}$ $\dot{x}$ x

$$m\ddot{x}+c\dot{x}+kx=F$$

$$m=5,\quad k=2,\quad c=1$$

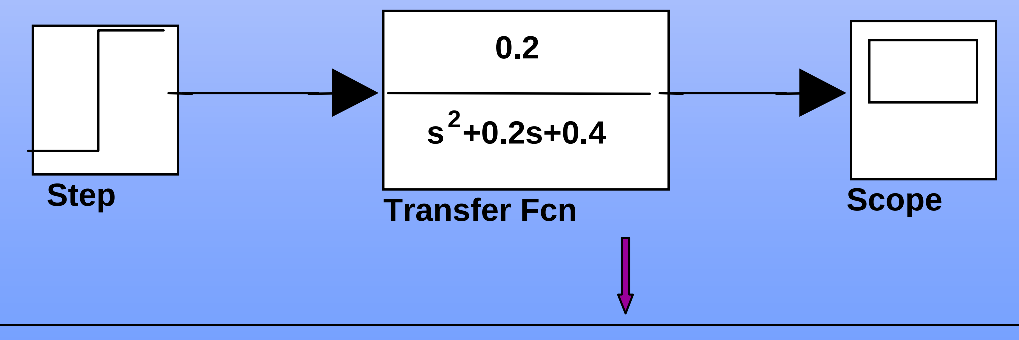
$$\ddot{x}=-0.2\dot{x}-0.4x+0.2F$$

$$G(s)=\frac{X(s)}{F(s)}=\frac{1}{ms^2+cs+k}$$

$$=\frac{1/m}{s^2+\frac{c}{m}s+\frac{k}{m}}$$

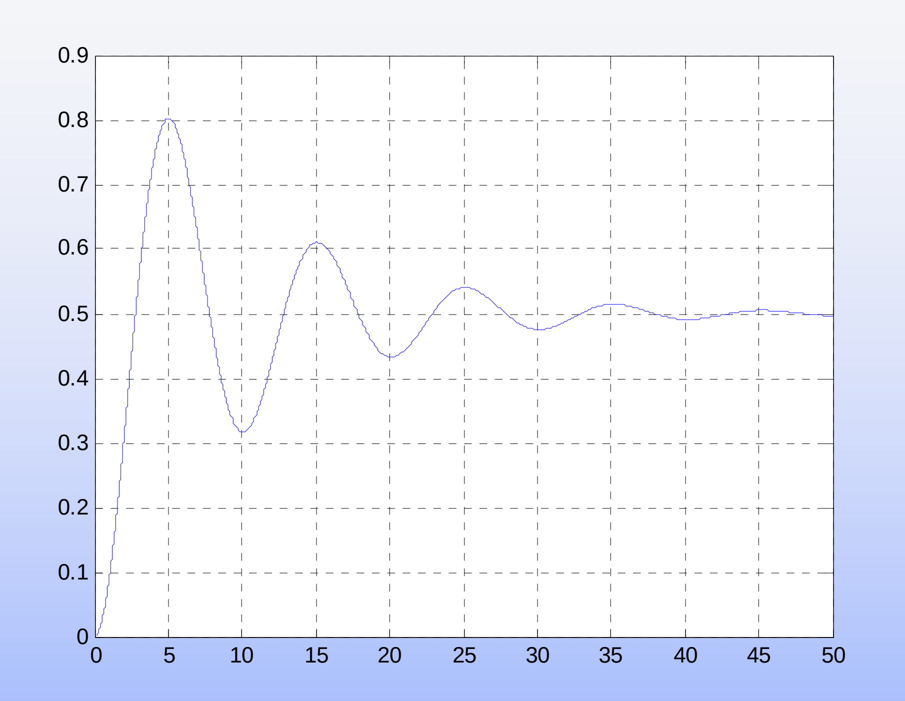
$$=\frac{0.2}{s^2+0.2s+0.4}$$

采用传递函数模块所建的模型

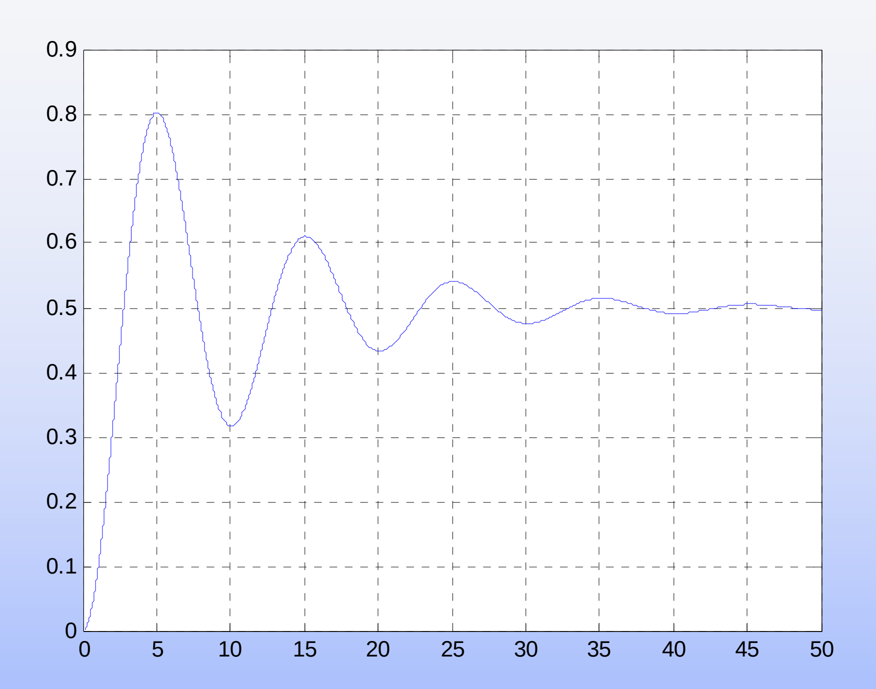


分子输入：0.2　　　分母输入：[1，0.2，0.4]

采用积分模块仿真的结果



采用传递函数模块仿真的结果



$$\ddot{x} = -0.2\dot{x} - 0.4x + 0.2F$$

采用幅值为**1**的阶跃函数激励，随着时间的推移，速度和加速度项逐渐趋于**0**，从而达到静平衡位置，**-0.4x＋0.2*1＝0**，即**x=0.5**

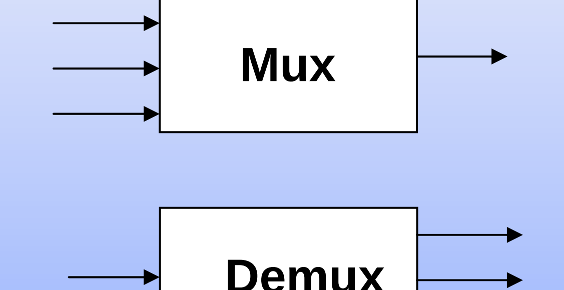
向量线性系统

信号线传输的信号可以是标量形式，也可以是向量信号形式。

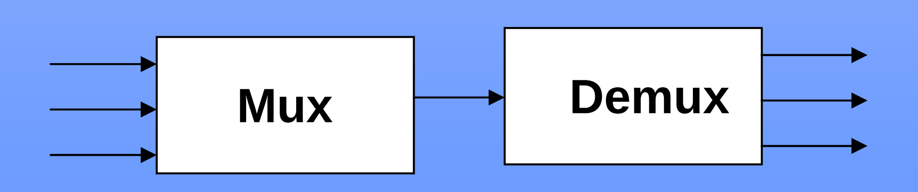
向量信号线

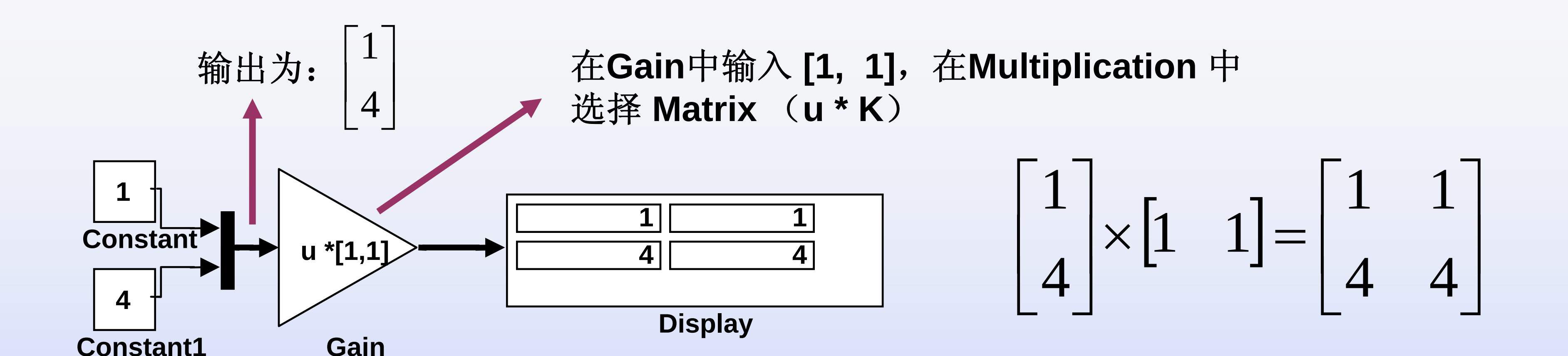
在 **Signals & Systems** 模块库中有一个 **Mux** 模块，它可以将多个标量信号合成为一个向量信号，输出为列向量。

与 **Mux** 相对应的模块是 **Demux** 模块，它用来将向量信号分解成一组标量信号。

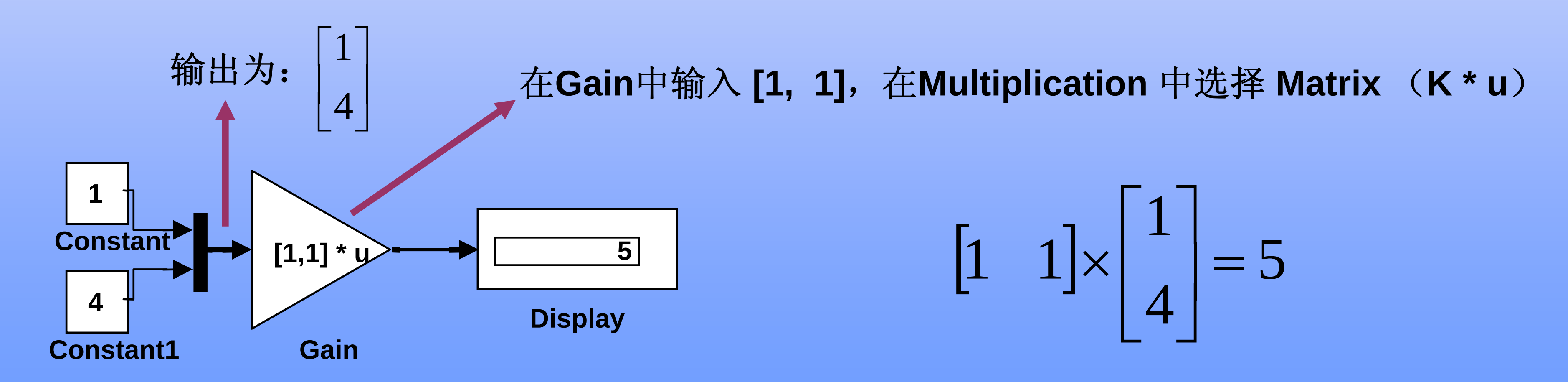


在 **Mux** 模块和 **Demux** 模块中必须正确指定输入或输出的个数。下图是将三个标量信号合成为一个向量信号，然后再分解为三个标量信号。

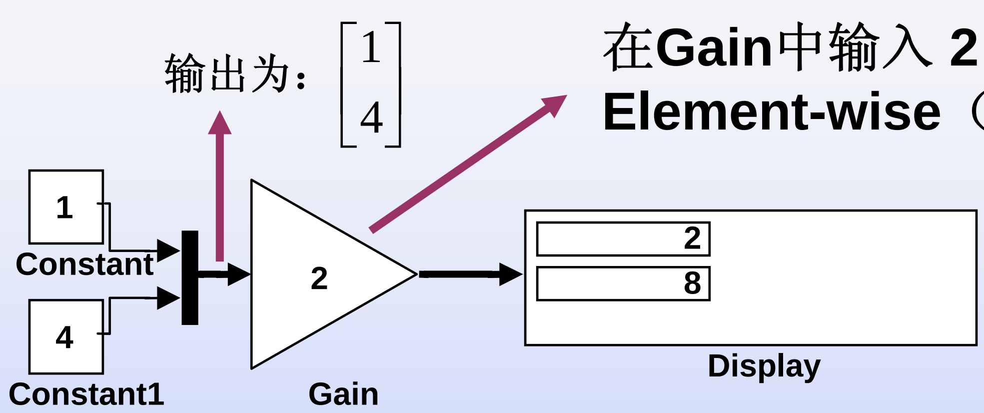




若在增益模块中选择 **Matrix (K * u)**，则结果如下。



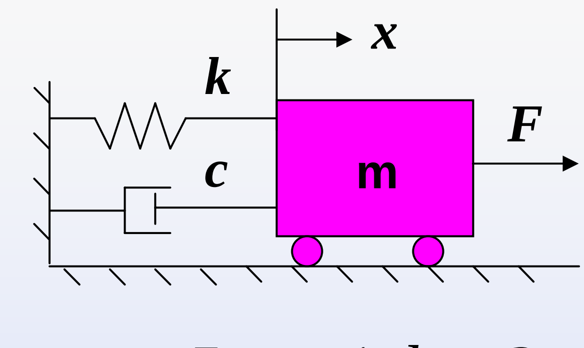
在**Gain**中输入 **2**，在**Multiplication**中选择**Element-wise（K. * u）**



$$\begin{bmatrix}1\\4\end{bmatrix}\times 2=\begin{bmatrix}2\\8\end{bmatrix}$$

状态空间模块

单自由度系统的单位脉冲响应



$$m\ddot{x}+c\dot{x}+kx=\delta(t)$$

$$5\ddot{x}+\dot{x}+2kx=\delta(t)$$

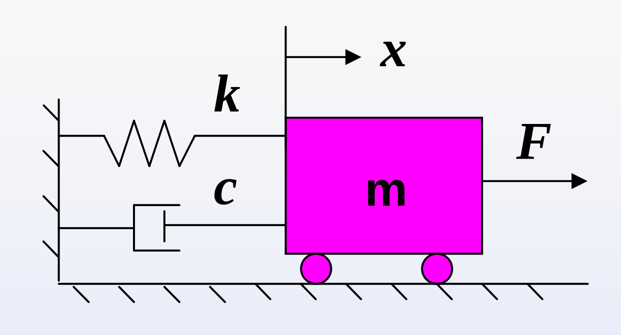
$$m=5,\ c=1,\ k=2$$

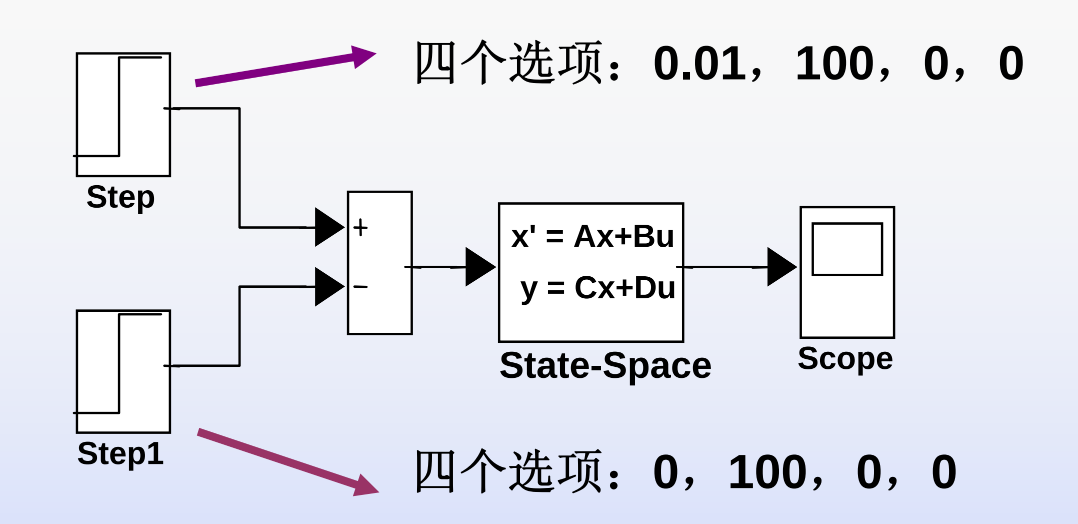
$$\boldsymbol{Y}=\begin{bmatrix}x\\ \dot{x}\end{bmatrix}\qquad \dot{\boldsymbol{Y}}=\begin{bmatrix}\dot{x}\\ \ddot{x}\end{bmatrix}=\begin{bmatrix}0 & 1\\ -0.4 & -0.2\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\end{bmatrix}+\begin{bmatrix}0\\ 0.2\end{bmatrix}\delta(t)=\boldsymbol{A}\boldsymbol{Y}+\boldsymbol{B}\delta(t)$$

输出量： $\boldsymbol{Z}=\boldsymbol{C}\boldsymbol{Y}$ 　　要求输出位移量，则：$\boldsymbol{C}=\begin{bmatrix}1 & 0\end{bmatrix}$

近似单位脉冲为两个有微小时间差的阶跃信号的差：

$$\delta(t)=100u(t)-100u(t-0.01)$$





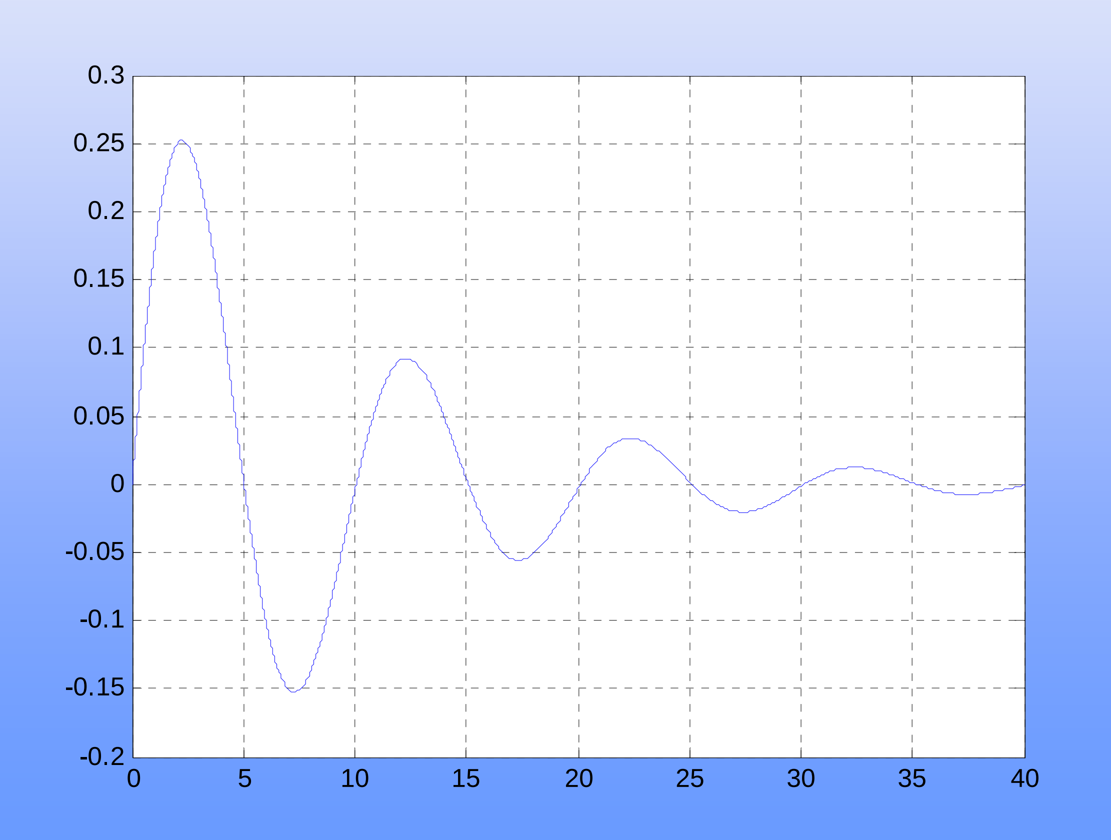
$$m\ddot{x}+c\dot{x}+kx=\delta(t)$$

$$m=5, c=1, k=2$$

$$\dot{\boldsymbol{Y}}=\begin{bmatrix}0 & 1\\ -0.4 & -0.2\end{bmatrix}\boldsymbol{Y}+\begin{bmatrix}0\\ 0.2\end{bmatrix}\delta(t)$$

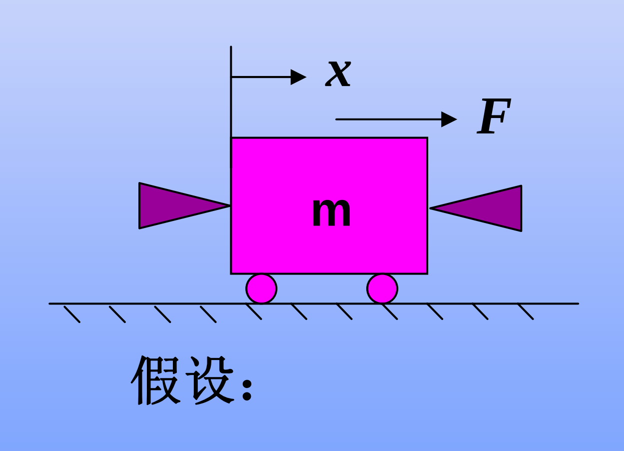
$$\boldsymbol{Z}=\boldsymbol{CY} \qquad \boldsymbol{C}=\begin{bmatrix}1 & 0\end{bmatrix}$$

$$\delta(t)=100u(t)-100u(t-0.01)$$



非线性系统的模拟

例子1：小车由两个喷射式发动机推动在光滑的平面内运动。若小车的速度和位移之和为负值，则启动左边的发动机；若小车的速度和位移之和为正值，则启动右边的发动机。控制的目标使小车静止在原点。此模型类似于卫星的位置控制过程。

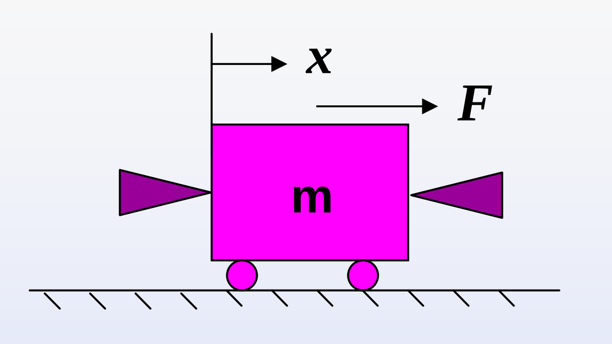


动力学方程： $m\ddot{x} = F$ $\quad \ddot{x} = \dfrac{F}{m}$

假设：

$$m = 5kg, \quad F = 1, -1, 0$$

$$\begin{cases} F = 1, & x + \dot{x} < 0 \\ F = -1, & x + \dot{x} > 0 \\ F = 0, & x + \dot{x} = 0 \end{cases}$$



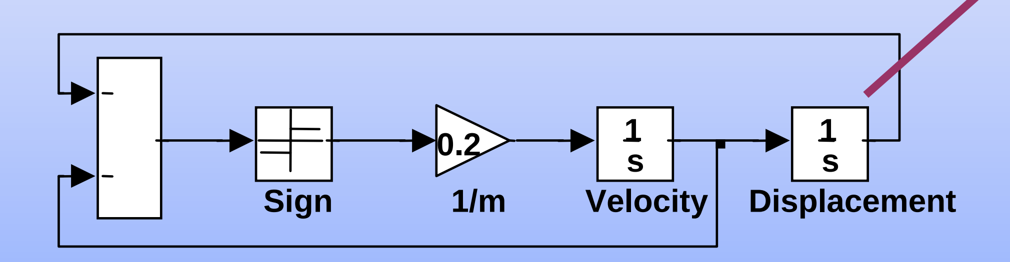
动力学方程： $m\ddot{x} = F \qquad \ddot{x} = 0.2F$

$$
\begin{cases}
F = 1, & x + \dot{x} < 0 \\
F = -1, & x + \dot{x} > 0 \\
F = 0, & x + \dot{x} = 0
\end{cases}
$$

假设： $m = 5kg, \quad F = 1, -1, 0$



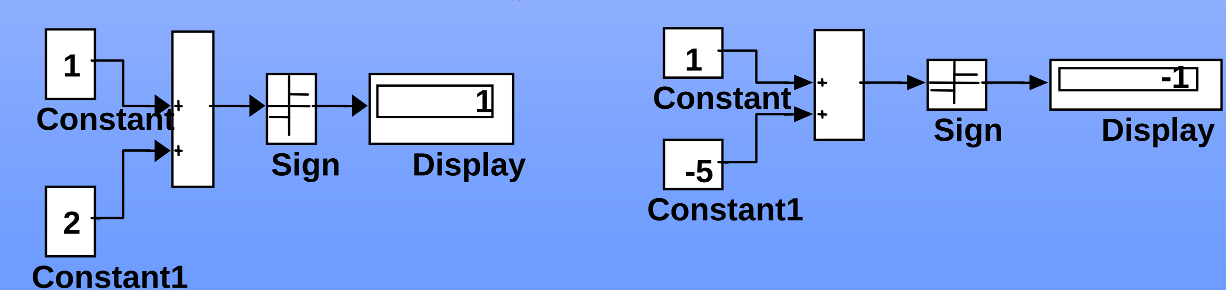
初始位移为＋1

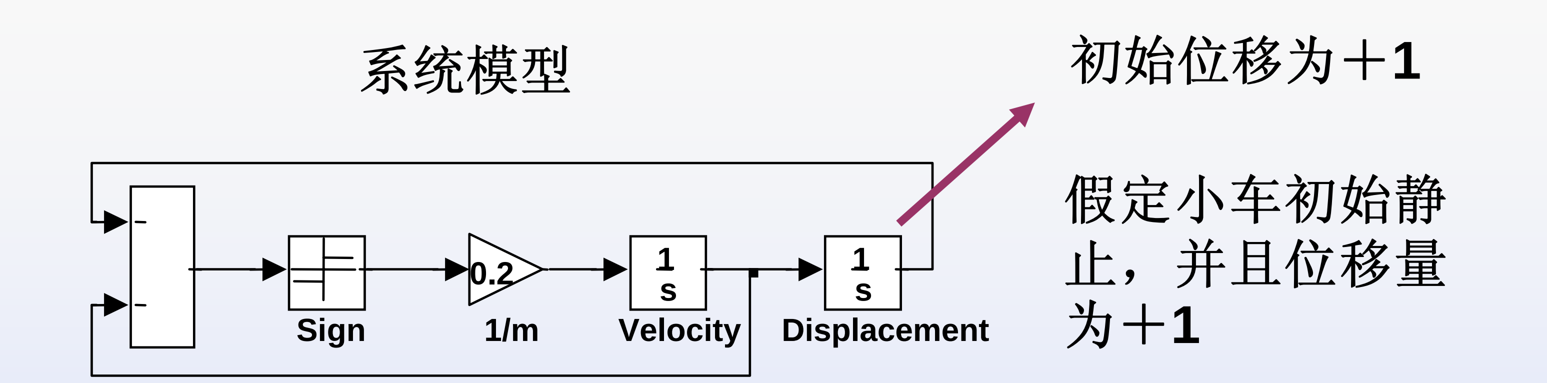


假定小车初始静止，并且位移量为＋1

$$
\begin{cases}
F = 1, & -x - \dot{x} > 0 \\
F = -1, & -x - \dot{x} < 0 \\
F = 0, & x + \dot{x} = 0
\end{cases}
$$

符号函数模块说明





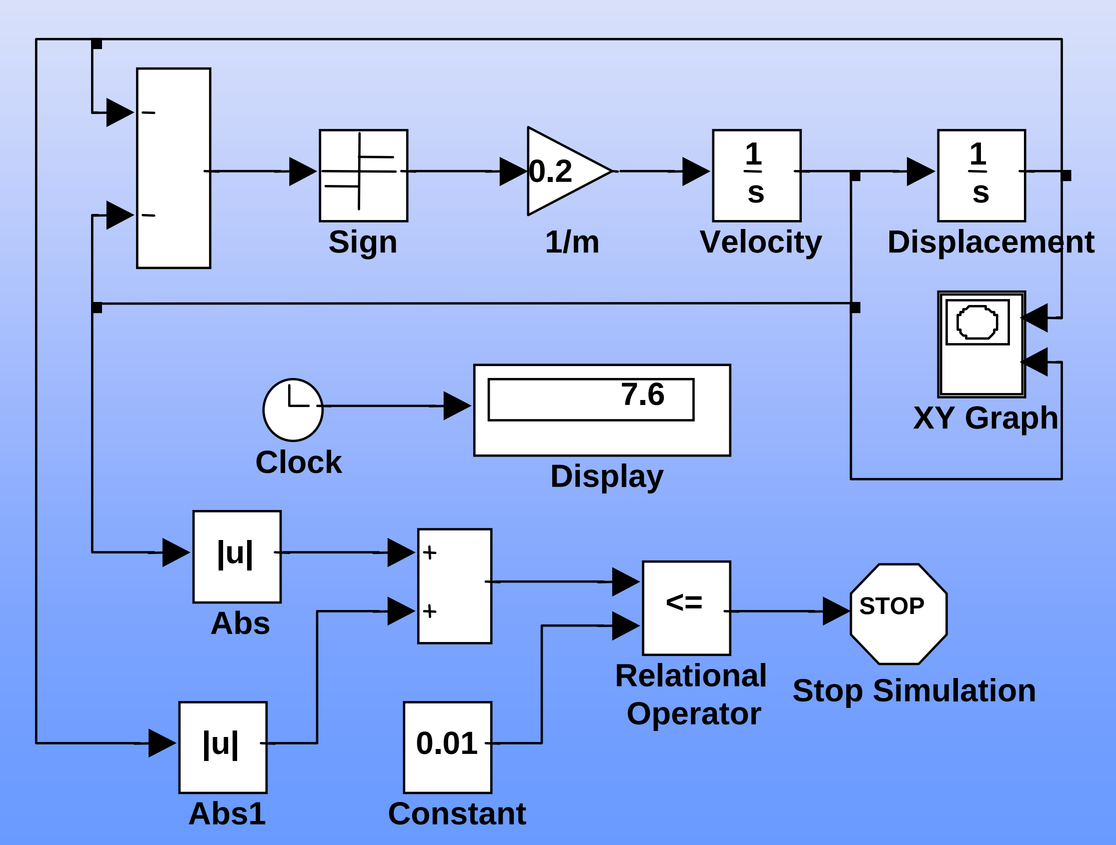
用一个二维图形模块来绘制仿真过程的相图。相图是速度相对于位移的变化图。为了观测时间，给模型加一个时钟模块。

注：在 **MATLAB** 中使用 **plot (xout(:,1), xout(:,2))** 画出的图形，和 **XY Graph** 的结果一致。

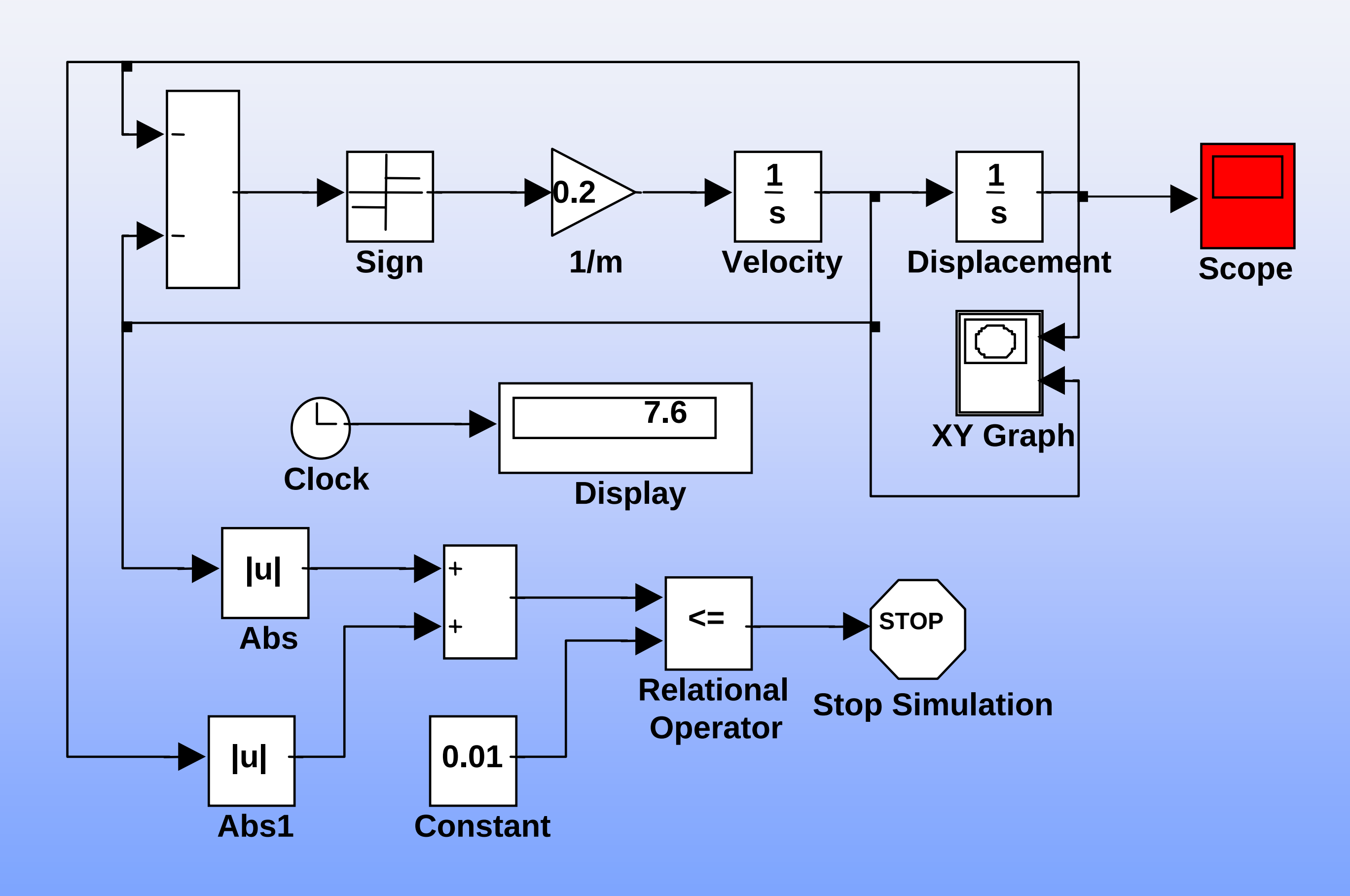
为了得知小车何时到达原点并使仿真停止，对模型增加一个逻辑判断，以使目标达到时仿真自动停止。

对于本算例，可以认为小车的位移和速度的绝对值之和小于一个较小量，如 **0.01** 时，就认为目标已经达到并结束仿真：

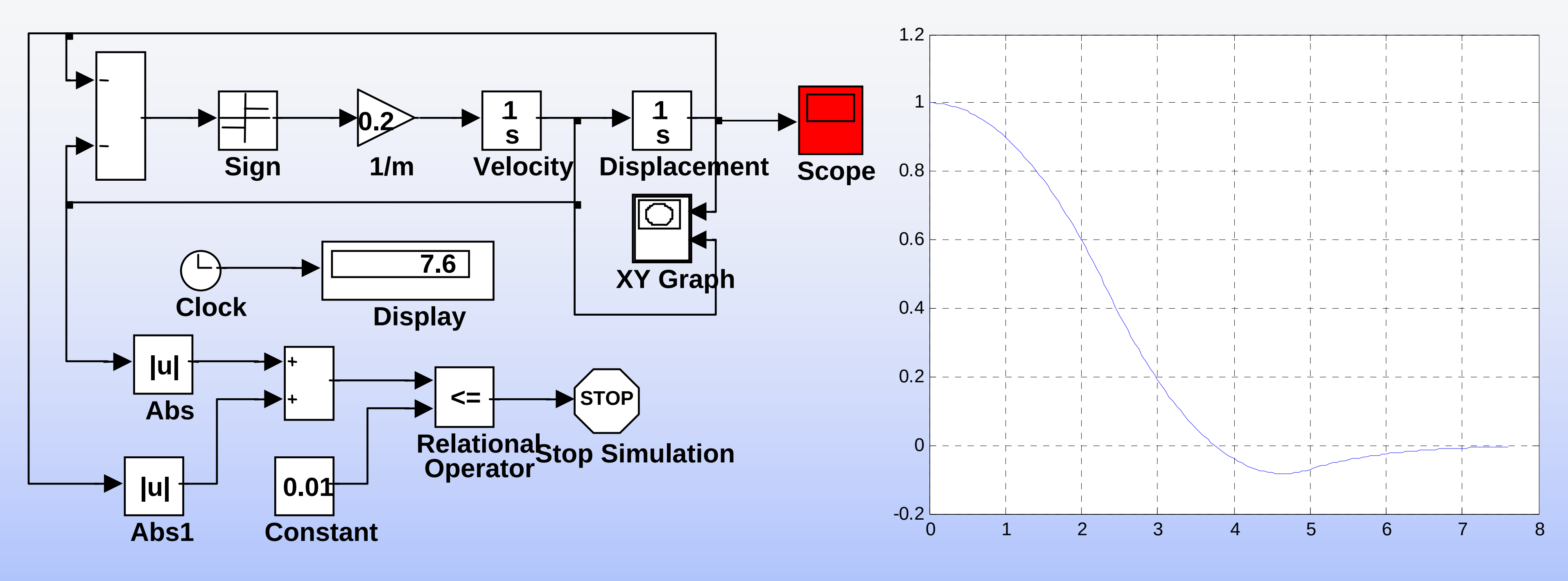
$$|x| + |\dot{x}| \leq 0.01$$



如果想得到小车位移的响应时程，可在模型中加入示波器模块，如下图所示：



仿真结果如下页图形所示。



该图形是采用 **MATLAB** 环境下的 **plot(tout, xout(:,1))** 画出的，结果与示波器的结果一致。由图形可看出，仿真在 **7.6 s** 时停止，这与模型框图中的时钟显示模块 **Display** 的结果一致。小车系统在 **7.6 s** 时达到平衡。

蹦极跳系统的动态仿真

蹦极跳时一种挑战身体极限的运动，蹦极者系着一根弹性绳从高处的桥梁（或山崖等）向下跳。在下落的过程中，蹦极者几乎处于失重状态。按照牛顿运动规律，自由下落的物体由下式确定：

$$m\ddot{x} = mg - a_1\dot{x} - a_2|\dot{x}|\dot{x}$$

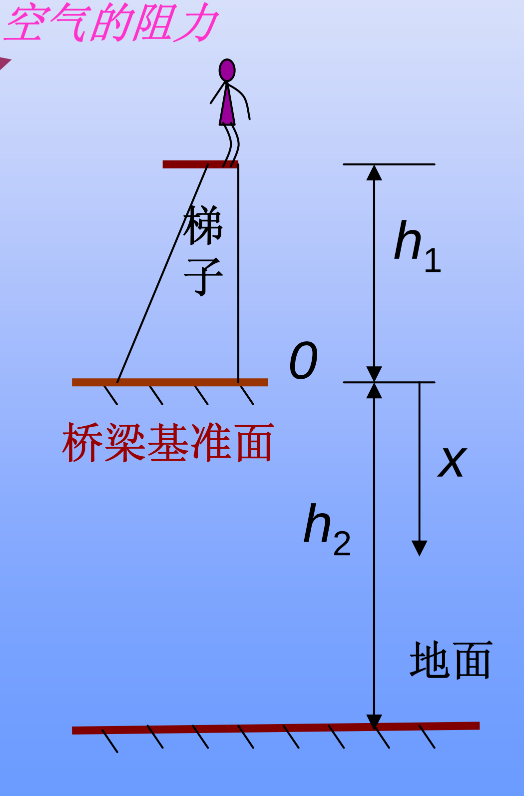
空气的阻力

m 为人体的质量　　　**g** 为重力加速度

位置 **x** 的基准为桥梁的基准面

如果人体系在一个弹性常数为 **k** 的弹性绳索上，定义绳索下端的初始位置为 **0**，则其对落体位置的影响为：

$$b(x) = \begin{cases} -kx, & x > 0 \\ 0, & x \le 0 \end{cases}$$



因此整个蹦极系统的数学模型为：

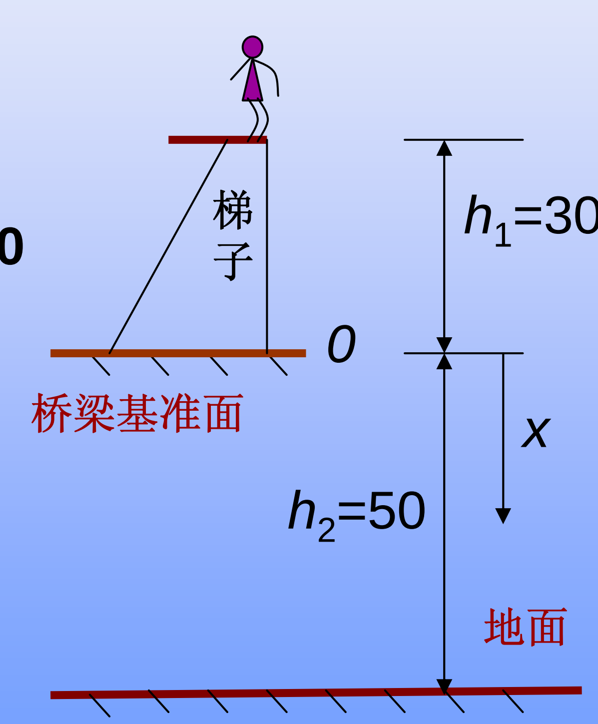
$$m\ddot{x} = mg + b(x) - a_1\dot{x} - a_2|\dot{x}|\dot{x} \qquad b(x) = \begin{cases} -kx, & x > 0 \\ 0, & x \le 0 \end{cases}$$

典型的具有连续状态的非线性系统

设桥梁距离地面为 **50 m**，即 h_2**=50**

蹦极者起始速度为 **0**

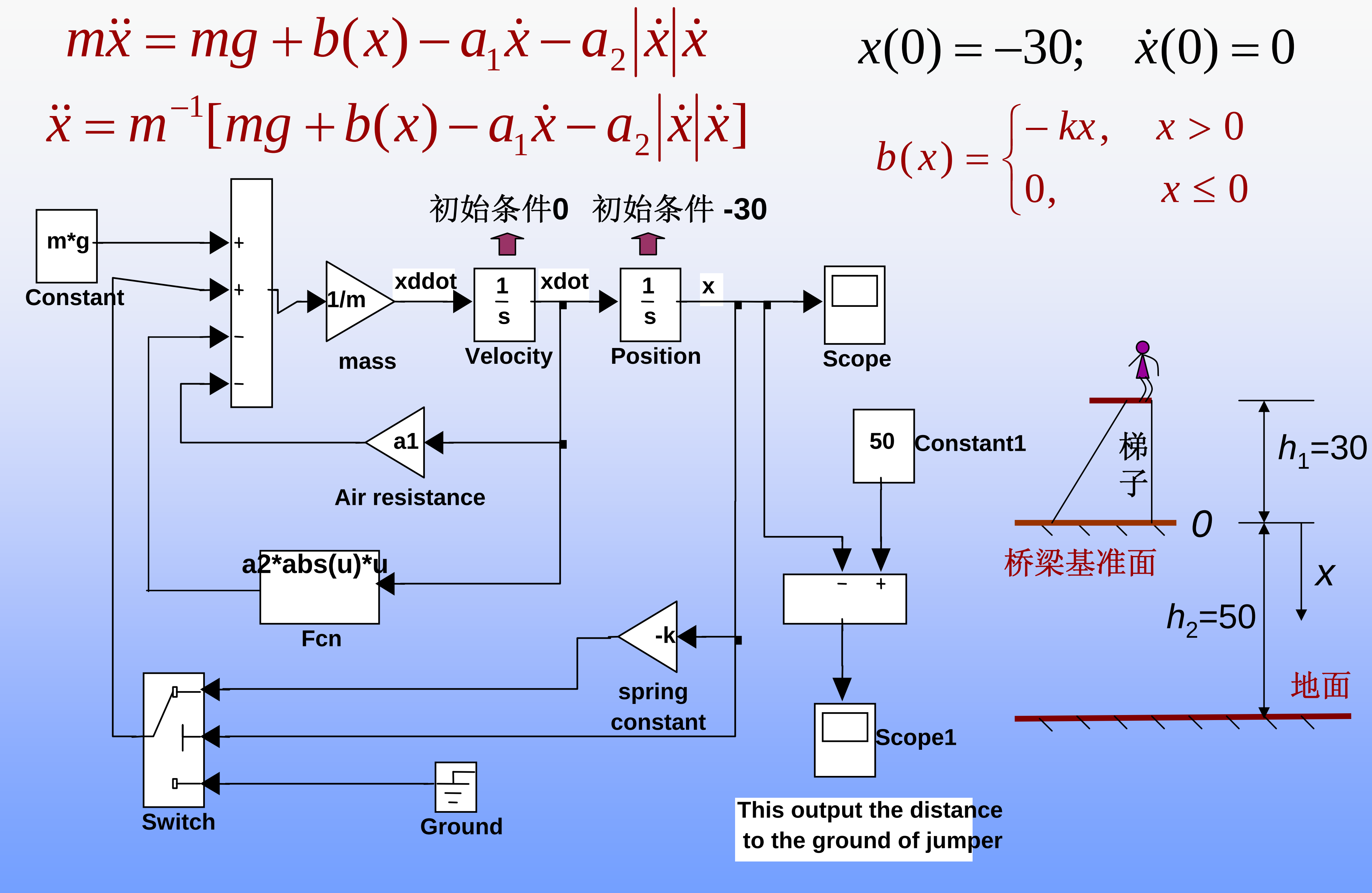
蹦极者的起始位置为绳索的长度 **30 m**，即 h_1**=30**



其余的参数：

$k=20$， $a_2=a_1=1$；$m=70$ **kg**，$g=10$ **m/s²**

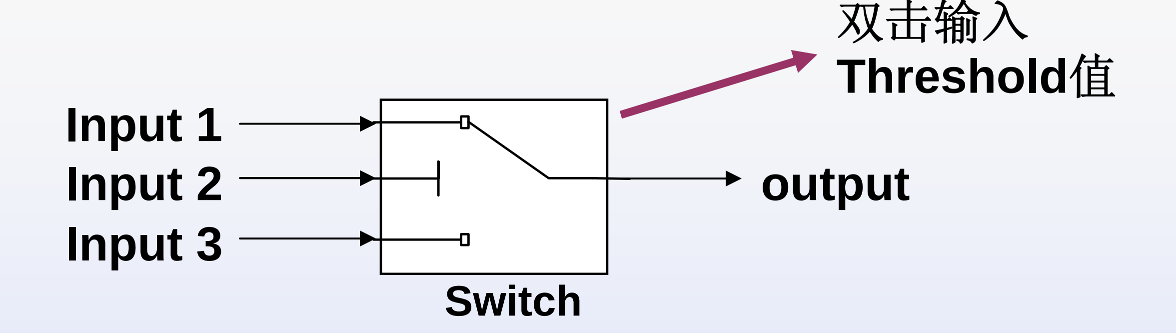
初始条件： $x(0) = -30; \quad \dot{x}(0) = 0$



说明：（1）设置仿真时间 0～100s。为了使曲线光滑，可设置最大仿真步长为 0.1。

（2）在 MATLAB 环境下输入 m=70；g=10；k=20；a1=1；a2=1；

Switch 模块介绍：

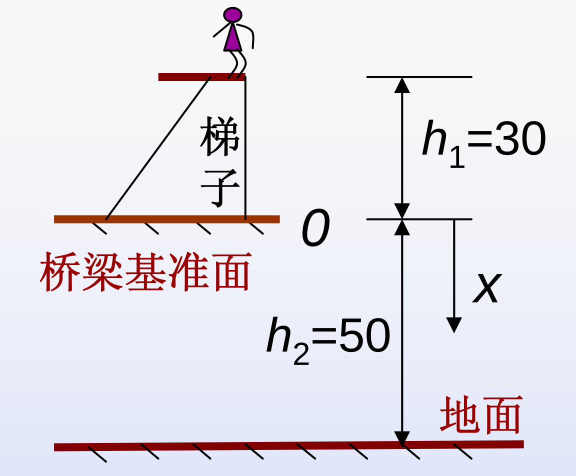


Pass through input 1 when input 2 is greater than or equal to threshold; otherwise, pass through input 3. The inputs are numbered top to bottom (or left to right)

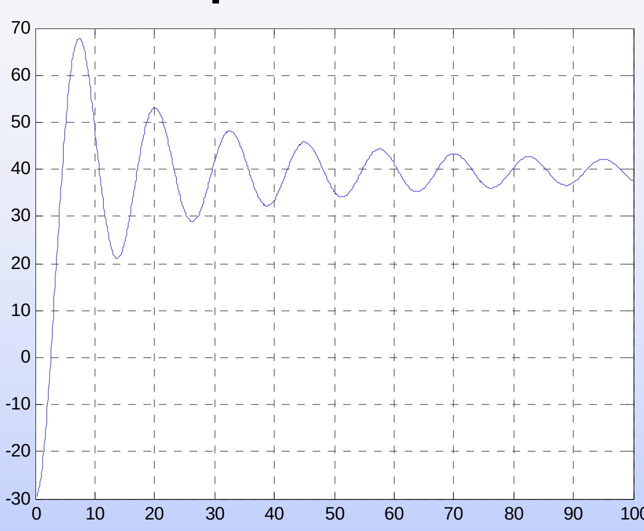
Threshold 默认为0

1
Constant
-10
Constant2
-1
Constant1
Switch
-1
Display

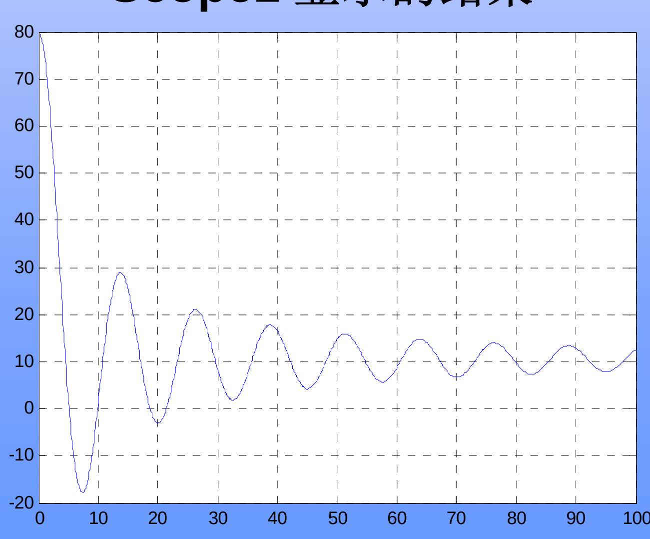
1
Constant
10
Constant2
-1
Constant1
Switch
1
Display



Scope 显示的结果







初始条件0 初始条件 -30

m*g

Constant

1/m

mass

xddot

1/s

Velocity

xdot

1/s

Position

x

Scope

a1

Air resistance

a2*abs(u)*u

Fcn

-k

spring constant

50 Constant1

Scope1

Switch

Ground

This output the distance to the ground of jumper

若在 **MATLAB** 环境下键入：**whos**

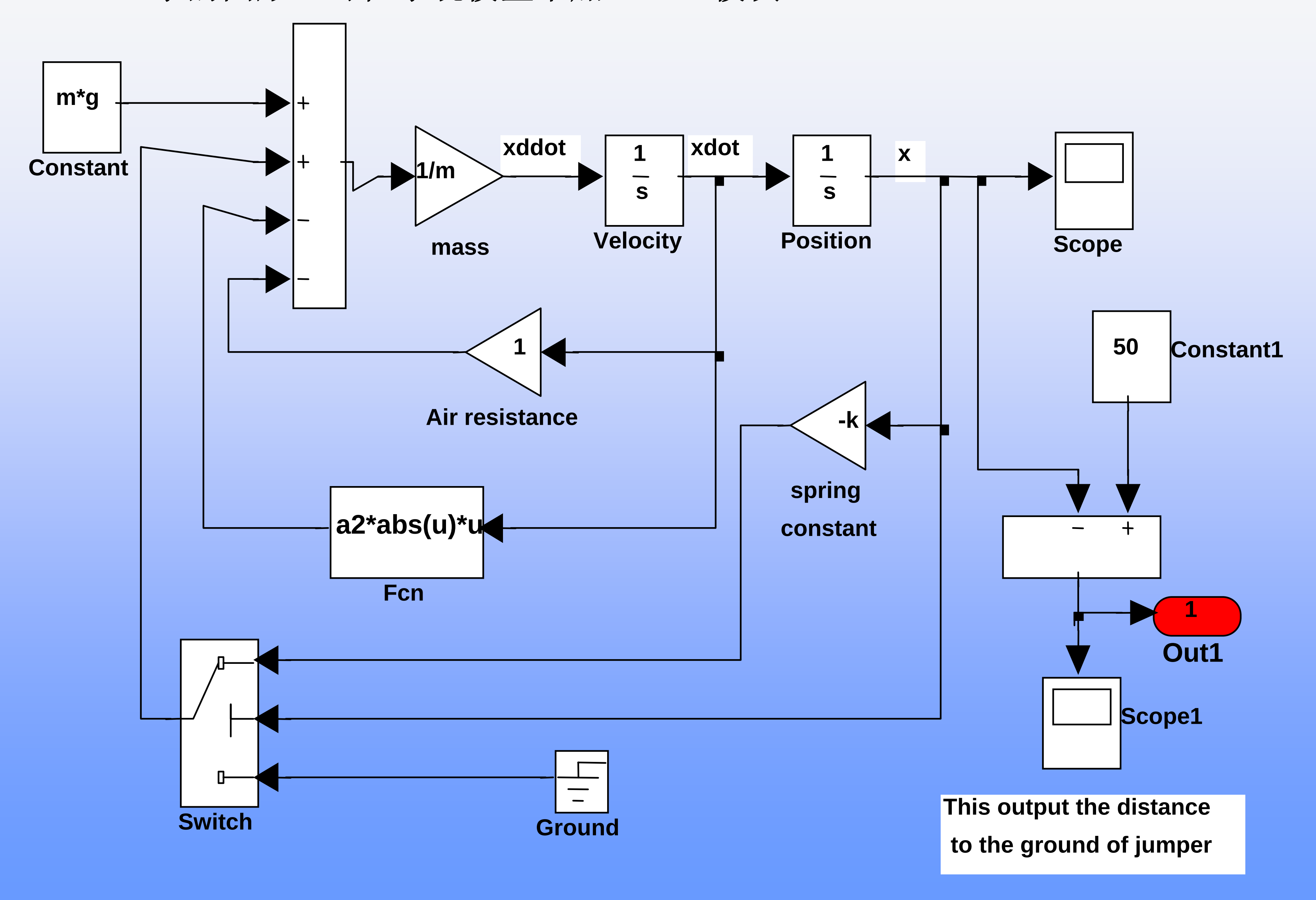
可得到：

whos

Name	Size	Bytes	Class
a1	1x1	8	double array
a2	1x1	8	double array
g	1x1	8	double array
k	1x1	8	double array
m	1x1	8	double array
tout	1000x1	8000	double array
xout	1000x2	16000	double array

tout 为时间序列；**xout** 第一列为位移时间序列，第二列为速度时间序列。若在 **MATLAB** 环境下键入：**plot (tout, xout (:, 1))**，将得到示波器 **Scope** 显示的结果

如果想在 **MATLAB** 环境下使用 **plot** 命令画出示波器 **Scope1**显示的图形，可在系统模型中加入 **out** 模块。



此时在 **MATLAB** 环境下键入：**whos**

可得到：

```
whos
  Name      Size          Bytes  Class

  a1        1x1             8  double array
  a2        1x1             8  double array
  g         1x1             8  double array
  k         1x1             8  double array
  m         1x1             8  double array
  tout   1000x1          8000  double array
  xout   1000x2         16000  double array
  yout   1000x1          8000  double array
```

其中 **yout** 为 **out** 模块的输出结果。此时若在 **MATLAB** 环境下键入：

plot (tout, yout)

将得到系统模型中示波器 **Scope1** 显示的结果。

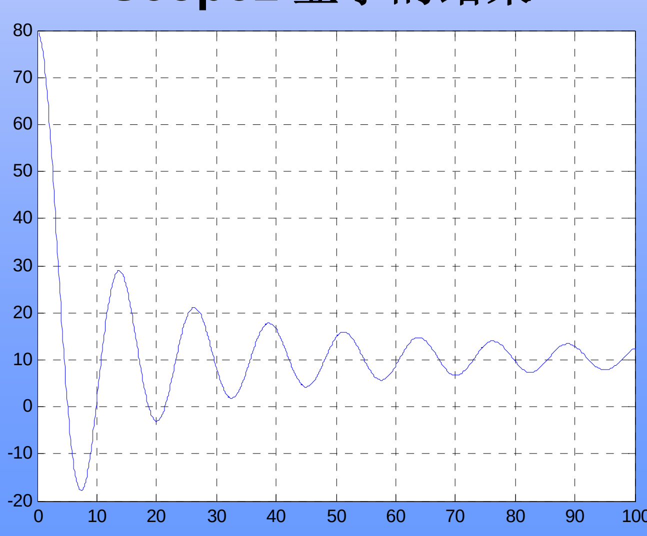
初始条件0　初始条件 -30

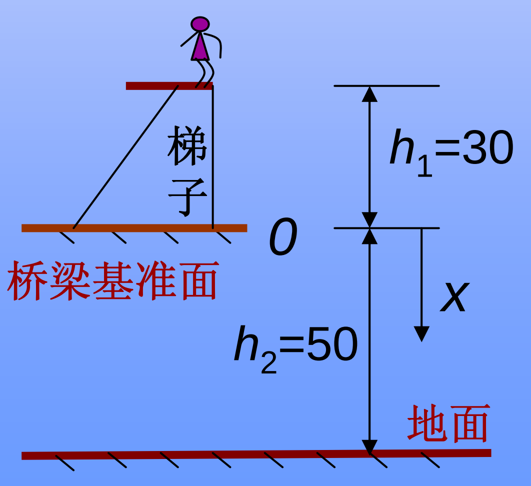
m*g
Constant
1/m
mass
xddot
$\frac{1}{s}$
Velocity
xdot
$\frac{1}{s}$
Position
x
Scope
a1
Air resistance
a2*abs(u)*u
Fcn
50
Constant1
-k
spring constant
Switch
Ground
Scope1

This output the distance to the ground of jumper









结果分析：

对于体重为 **70 kg** 的蹦极者，此系统是不安全的，因为蹦极者与地面之间的距离出现了负值

SIMULINK (6)

子系统及其封装技术

本章学习内容和目的

- 掌握 Simulink 子系统的建模方法
- 掌握 Simulink 子系统的封装技术

在前面的章节中，介绍了使用 **Simulink** 进行建模的基本方法。使用这些方法基本可以创建任何物理系统的模型。然而随着系统越来越复杂，用这些基本操作创建的 **Simulink** 模型变得越来越庞大而难于读懂。在本章中，将介绍一系列的 **Simulink** 的特殊处理技术，使得模型变得更加简捷和易懂易用。

本章首先介绍一种类似于程序设计语言中的子程序的处理方法——**Simulink** 子系统，然后讲解一种更加好用的封装子系统的技术。

Simulink 子系统

绝大多数程序设计语言都有使用子程序的功能，例如 **MATLAB** 的子程序——**M**文件。随着模型变得越来越大、越来越复杂，人们很难轻易读懂它们。在这种情况下，子系统通过把大的模型分割成几个小的模型以使得整个模型更加简捷、可读性更高，而且这种操作并不复杂。

创建 **Simulink** 子系统有两种方法：

（**1**）对已经存在的模型的某些部分或全部使用菜单命令**【Edit>Creat Subsystem】**将你性压缩转换，使之成为子系统；

（**2**）使用 **Subsystems** 模块库中的 **Subsystem** 模块直接创建子系统。

Simulink 子系统的两种作用：

（**1**）系统模型更加简捷和可读性高；

（**2**）子系统可以反复调用，节省建模时间。

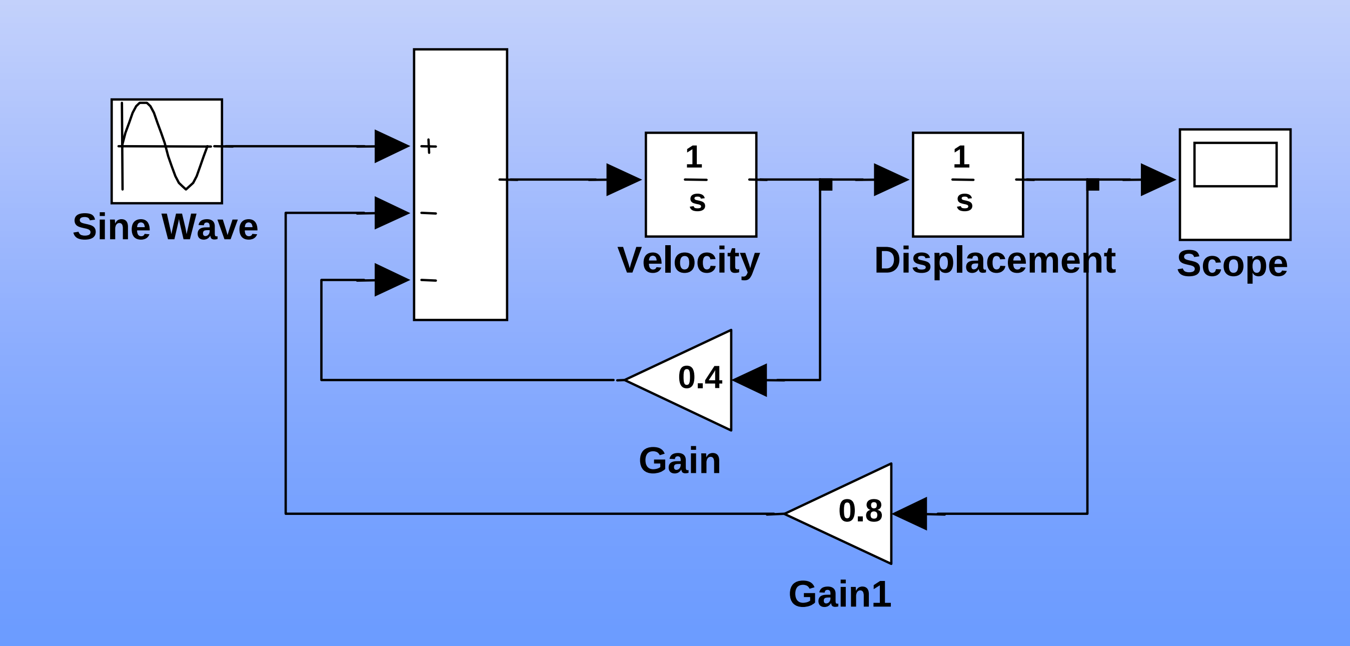
压缩子系统

以一个例子说明压缩子系统的使用方法。例如：

$$\ddot{x} + 0.4\dot{x} + 0.8x = \sin(t)$$

方程可转化为：

$$\ddot{x} = -0.4c\dot{x} - 0.8x + \sin(t)$$

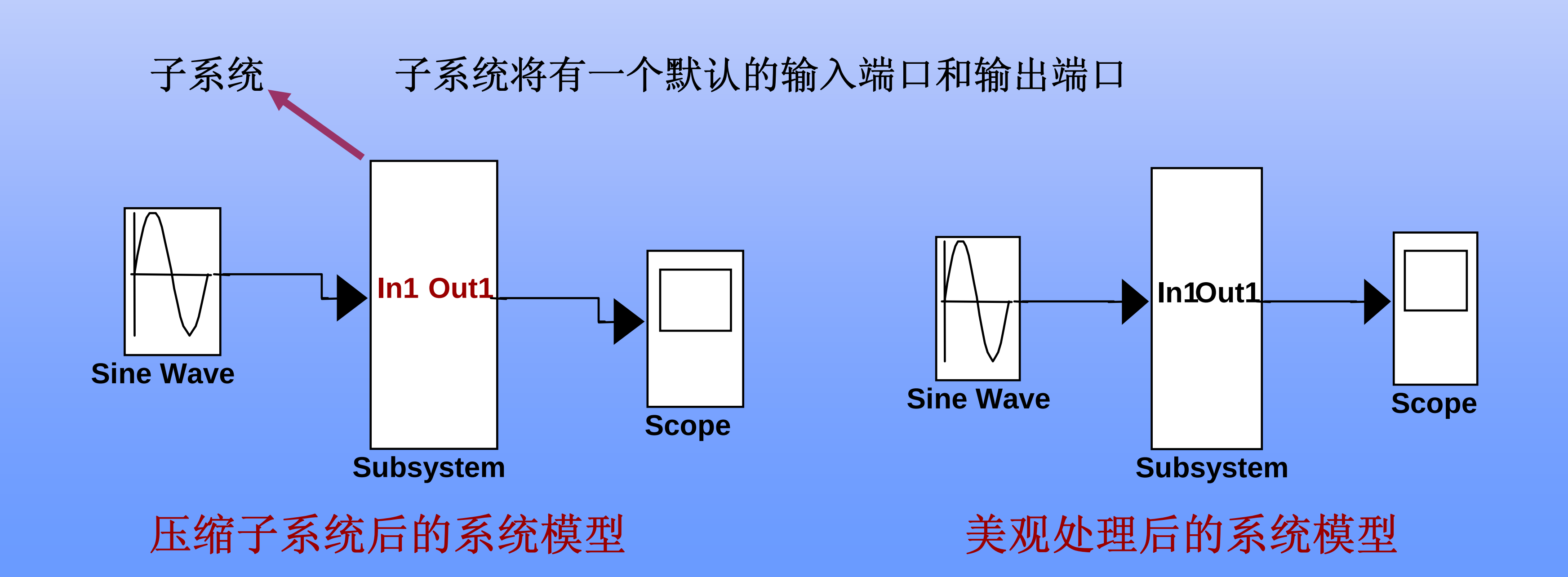


操作步骤：

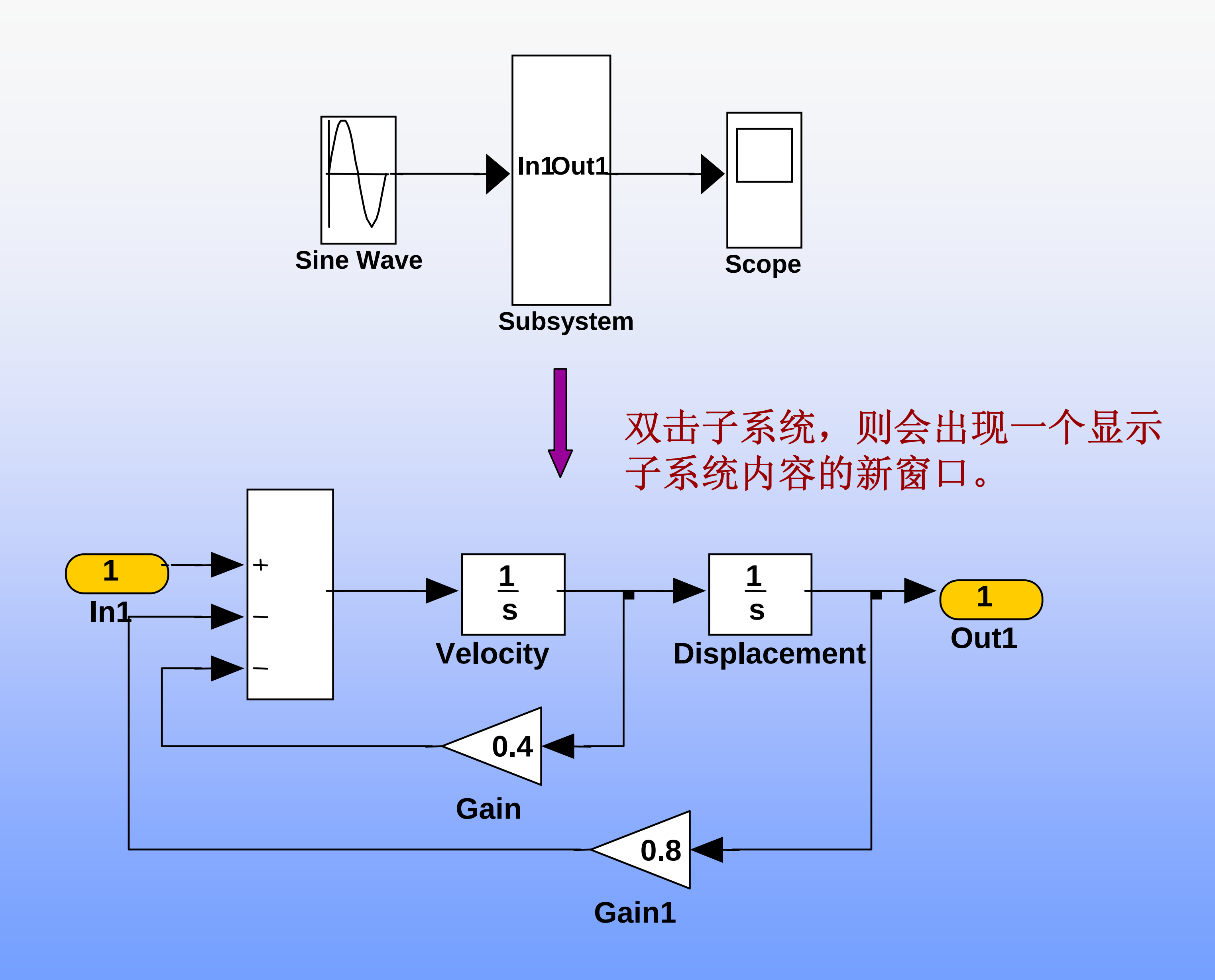
（1）使用范围框将要压缩的子系统的部分选中，包括木块和信号线；（注意：只能使用范围框，而不能使用 **Shift**逐个选定）

（2）在模块窗口选项中选择**【Edit>Creat Subsystem】**，**Simulink** 将会用一个子系统模块代替被选中的模块组；

（3）进行模型美观调整。



压缩子系统后的系统模型　　　　美观处理后的系统模型



在新窗口中，除了原始的模块外，**Simuink** 自动添加了输入模块和输出模块，分别代表子系统的输入端口和输出端口。

两点说明：

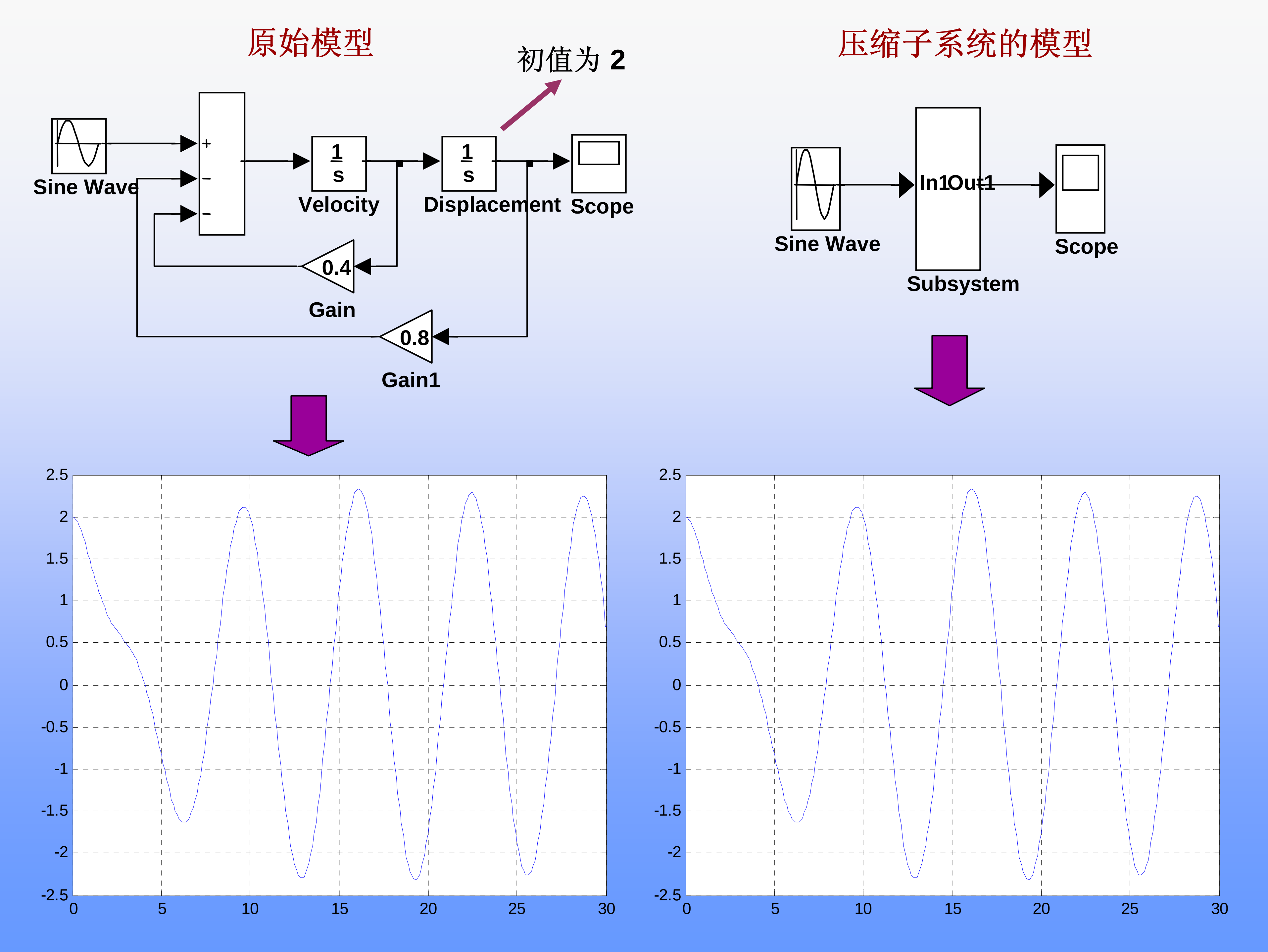
（**1**）子系统窗口无需保存，只需保存主程序出口即可。保存主程序窗口后，子系统窗口自动得以保存；

（**2**）菜单命令 **【Edit>Creat Subsystem】** 没有相反的操作命令，也就是说，一旦一组模块压缩成了子系统，就没有可以直接还原的处理方法（**undo** 除外）。因此，一个理想的处理方法是在压缩子系统之前，先将模型进行保存，作为备份。

以上例子的运行结果

假定系统有如下初始条件： $x(0)=2, \quad \dot{x}(0)=0$

要求采用原始模型（没有压缩子系统的模型）和压缩子系统的模型进行仿真，仿真时间**30s**。对比结果。

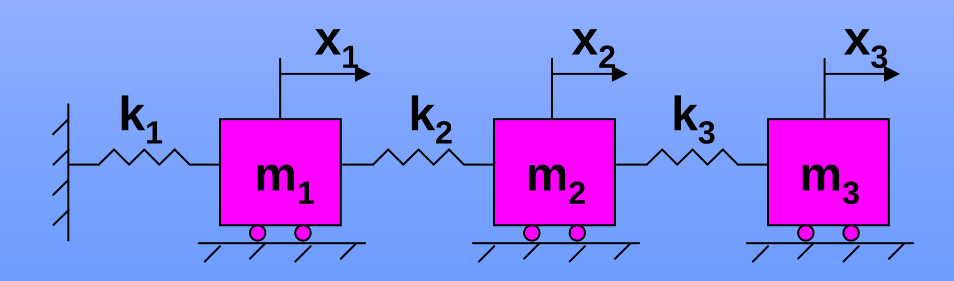

原始模型
压缩子系统的模型
初值为 2
Sine Wave
+
−
−
1
s
Velocity
1
s
Displacement
Scope
0.4
Gain
0.8
Gain1
Sine Wave
In1
Out1
Subsystem
Scope
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
5
10
15
20
25
30
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
5
10
15
20
25
30

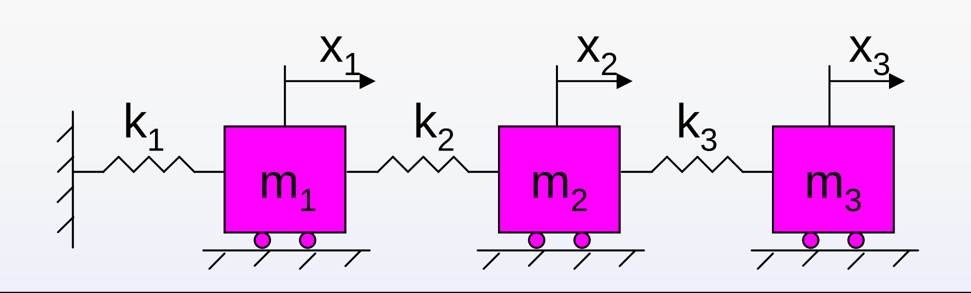
子系统模块

在创建模型的时候，如果需要一个子系统，除了上述介绍的压缩子系统的方法外，也可以直接使用子系统模块，在子系统窗口中进行创建。

要使用子系统模块创建子系统，先从 **Subsystems** 模块库中拖曳一个子系统模块到模型窗口中，然后双击子系统模块，就会出现一个子系统的编辑窗口，子系统的建立可以在该窗口中进行建立。

例子：模拟下图所示的弹簧－质量系统的运动状态。





$m_1\ddot{x}_1$

$k_1 x_1$ m1 $k_2(x_1 - x_2)$

$$m_1\ddot{x}_1 + k_1 x_1 + k_2(x_1 - x_2) = 0$$

$$\Rightarrow \quad \ddot{x}_1 = \frac{1}{m_1}\left[k_1(-x_1) + k_2(x_2 - x_1)\right]$$

$k_2(x_1 - x_2)$ m2 $k_3(x_2 - x_3)$

$$m_2\ddot{x}_2 - k_2(x_1 - x_2) + k_3(x_2 - x_3) = 0$$

$$\Rightarrow \quad \ddot{x}_2 = \frac{1}{m_2}\left[k_2(x_1 - x_2) + k_3(x_3 - x_2)\right]$$

$m_3\ddot{x}_3$

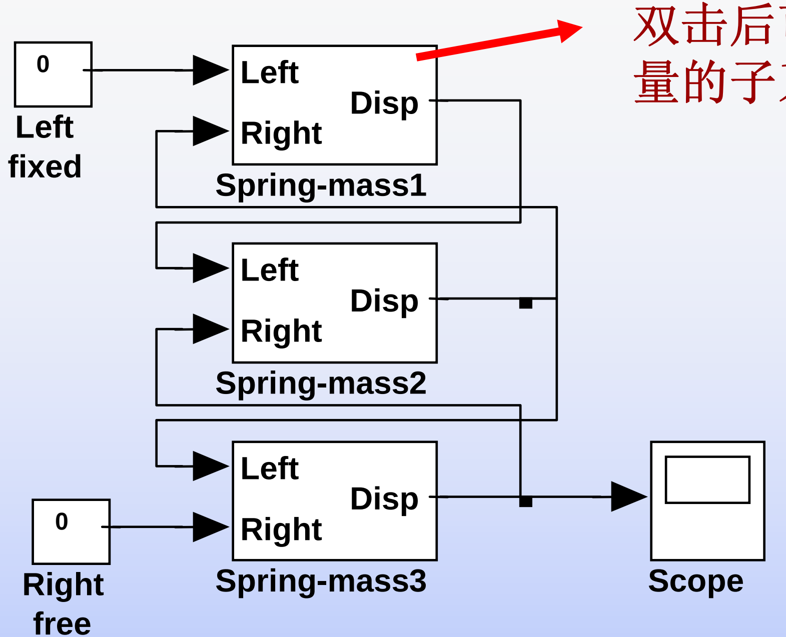
$k_3(x_2 - x_3)$ m3

$$m_3\ddot{x}_3 - k_3(x_2 - x_3) = 0$$

$$\Rightarrow \quad \ddot{x}_3 = \frac{1}{m_3}\left[k_3(x_2 - x_3)\right]$$

单个质量的运动方程：

$$\ddot{x}_n = \frac{1}{m_n}\left[k_n(x_{n-1} - x_n) + k_{n+1}(x_{n+1} - x_n)\right]$$

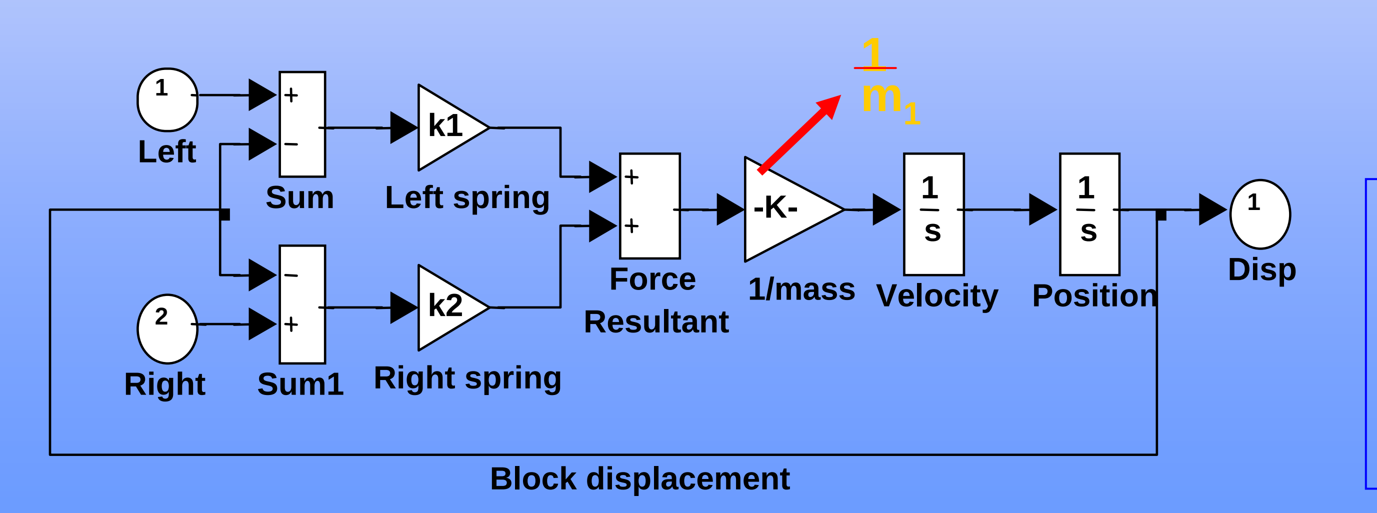


双击后可显示出第一个质量的子系统图，下图所示

第二和第三个质量子系统中，三个增义模块分别输入：

第二个质量：K_2，K_3，$\frac{1}{m_2}$

第三个质量：K_3，0，$\frac{1}{m_3}$



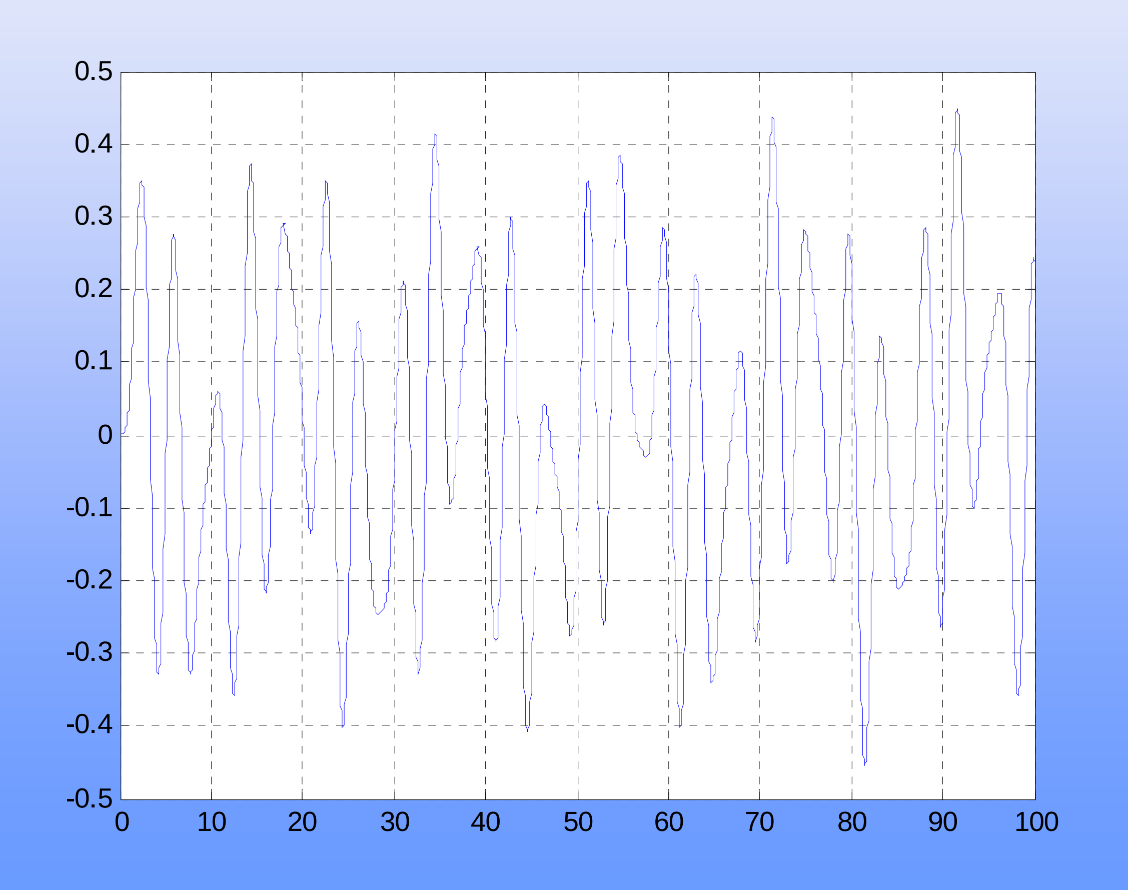
$\frac{1}{m_1}$

注意：在子系统的信号输入端要使用一个输入模块，在信号输出端要使用一个输出模块。

假定系统的初始条件为：第一个质量上有初始位移 $x_1(0)=1$，则在第一个质量子系统框图中的位移模块中输入初始条件 1。 在 MATLAB 环境下输入：

K1=1; K2=2; K3=4; m1=1; m2=3; m3=2

系统仿真结果如下：



说明：若想输出其它两个质量的轨迹，可在模型中增添示波器模块 **Scope**，或增添输出模块 **Out** 即可。

封装模块

封装技术是将 **Simulink** 子系统“包装”成一个模块，并可以如同使用 **Simulink** 内部模块一样使用的一种技术。每个封装模块都可以有一个自定义的图标和一个用来设定参数的对话框。参数的设定方法也与 **Simulink** 模块库中的内部模块完全相同。

作业：

封装蹦极系统。要求：封装后的蹦极子系统只有一个输出端口，封装后子系统的参数设置包括蹦极者的体重、弹性绳索的弹性常数。通过仿真分析蹦极系统在下述情况下是否安全，并绘制响应的响应曲线：

（**1**）蹦极者体重 **80 kg**，弹性绳索的弹性常数为 **30**；

（**2**）蹦极者体重 **70 kg**，弹性绳索的弹性常数为 **20**。